Master Thesis

Restaulytics
Sentiment Analysis in Restaurant Reviews

Author:

Ștefan Georgiana-Denisa
Master in Innovation and Research in Informatics
Facultat d'Informatica de Barcelona
Universitat Politècnica de Catalunya

Advisor:

Ms. Dr. Nuria Castell Ariño
Natural Language Processing Group
IDEAI Research Center
Departament CS
Universitat Politècnica de Catalunya

Co-Advisor:

Conf. Dr. Eng. Mihai Dascălu
Department of Computer Science
Faculty of Automatic Control and Computers
University Politehnica of Bucharest

2019

ABSTRACT

This thesis aims to investigate the problem of sentiment analysis and how it can be solved as a 5 classes classification problem with automatic learning. This is done via experimenting with different types of features and models. The experiments are centred around the restaurant domain with classes derived from the number of stars assigned by users in reviews. These stars summarize the users' appreciation of their experience in a restaurant on a positivity scale from 1 to 5. It is shown that, out of all the models designed, the convolutional network that allows for the GloVe continuous representation of words to be adjusted in response to backpropagation reports the highest performance – 67.82% accuracy on the fine-grained classification. This result supports the hypothesis that transfer learning and focusing on local patterns, techniques that have proven very helpful in computer vision problems, are helpful as well when deducting meaning from the natural language discourse. Moreover, it is shown that this automatic learning of features leads to better results than the proposed manual representations (handcrafted features) of reviews that were trying to leverage sentiment associated with topic information. Interest was also shown to the simpler classifiers such as Naïve Bayes or Support Vector Machines, which acted as baseline methods, but yet performed better that some feed-forward neural networks. These results clearly show that adding complexity does not guarantee added performance and that the particularities of the problem need to be taken into account.

Table of Contents

List of Figures

List of Tables

1. Introduction

At first, the term “Restaulytics” may seem peculiar. However, as a combination of the two words: “restaurant” and “analytics”, it perfectly summarizes the content of this thesis, which is the result of the research effort of designing models that could automatically analyse and extract sentiment information from restaurant reviews in natural language form. Thus, it proposes forms of supervised and unsupervised machine learning together with natural language processing techniques in order to find how a restaurant is perceived by a client in terms of number of stars. At the same time, it dives deeper into the reasons why an approach works better than the other. Such an effort is well justified and motivated by the practical applications of such models.

1.1. Motivation

Market research is quintessential to the marketing decisions of an organization. This comes as a natural result, given that consumers are at the centre of the marketing activity and research offers the means to study their perceptions and behaviour. These means can be either quantitative or qualitative. Usually, a quantitative research implies distributing questionnaires to a representative sample of people and statistically interpret the results. Qualitative research, on the other hand, is not about numbers and figures, but about exploring or gaining a more in-depth perspective on a topic through focus-groups and/or in-depth interviews. It can be a preliminary step of the quantitative research, setting its premises, but also a subsequent one, as a means of finding out the motives behind the statistical results.

Nowadays, in the context of Web 2.0, the process of gathering data for research can be not only “push-based”, but also “pull-based”, as users can freely and publicly express their opinions in the online medium. The information is already there, the organisation only needs to gather and process it. However, it’s hard to make sense of such a quantity of unstructured information. An online review comes in natural language form and it is much like a mini in-depth interview. But in a classic research, there are only a few people interviewed. Here, the organisation deals with tens, hundreds or even thousands of people expressing opinions, making up the representative sample required in quantitative research, while keeping the in-depth information specific to qualitative research. However, in order to get the same statistical results that we would get in a quantitative research, we need a way to structure the data, using a common template for all reviews. This is where natural language processing (NLP) and machine learning (ML) come in hand. NLP

techniques offer the means to structure the data and extract meaningful features from it, while ML discovers the patterns behind these features. This is all done automatically, requiring a much smaller monetary cost than a classical study. Moreover, once the models are up and running, the results can be gathered in real time. There is no longer a delay between a phenomenon happening, an organisation observing it and ordering a research, data being gathered, metrics computed, and results interpreted and delivered.

1.2. Objectives

The main objective of this thesis is to explore the problem of sentiment analysis in a specific domain, which is the restaurant domain, as a classification problem with 5 classes (from 1 to 5 stars) and also go one step further and try to find the reasons why certain approaches and models work better than others. This objective is accomplished mainly by means of experimentation with different type of models and different types of recipes for pre-processing the reviews so as to deriver features for the selected models and improve accuracy.

2. State of the art

2.1. Definition

Sentiment analysis is a research area that aims to identify opinions or sentiments, together with their respective targets, as expressed directly or indirectly in text. In much more detail, the essence of the domain can be understood from the problems it tries to solve:

- Given a text that expresses an opinion of an object, find the polarity of the opinion, either by identifying it with one of the two opposed polarities (classification) or by placing it on the scale between those two polarities (regression).
- Classify a text as subjective or objective.
- Identify the topics in a text and the opinions associated.
- Recognize humour, irony and sarcasm.
- Find the author of a text. (Pang & Lee, 2008)

2.2. Short history

"Sentiment analysis" as a term appeared for the first time in the article "Sentiment analysis: capturing favorability using natural language processing" published in 2003 by the IBM researchers Nasukawa and Yi. The research field is also known as "opinion mining", term that appeared in the same year in the article "Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews" published by Dave et al. Other names include: opinion extraction, sentiment mining, subjectivity analysis (Jurafsky, 2018). The last one is justified by the observation that sentences expressing opinions are inherently subjective (Liu, 2015).

Given that most of the work so far focused on the written text, sentiment analysis is considered a subfield of the NLP. Although previous work existed, the field started to gain popularity and interest from the research world at the beginning of this century, due to several factors. One is the emergence of the social web, which led to an abundance of opinions saved in digital format that can be used as data by the researchers in this field. The other is the development and use of automatic learning algorithms in NLP and information retrieval.

Existing applications of sentiment analysis include: understanding consumer feedback, review summarization, brand analysis and reputation management, finding the emotional state of a nation, public opinion understanding, event monitoring, media studies, and financial predictions

(there are studies stating the correlation between a nation's emotional state and stock price changes) (Potts, 2011).

2.3. Problem formalization

As stated earlier, sentiment analysis aims to identify sentiments from written texts. Thus, the following question arises naturally: What are those sentiments? Psychology differentiates between affect, emotion and mood. The affect is a neuropsychological state consciously accessible as the simplest brute, unprocessed, raw feeling. It is obvious in both emotions and moods. Emotion is a more complex feeling, indicator of an affect, but targeted towards an object, a person, a subject, or an event. It has high intensity and lasts for a short period of time. In comparison, mood is a more diffuse feeling, it is not as focused on a target, it has lower intensity and it lasts longer (Russell, 2003). Using this distinction, Liu (2015) states that researchers in sentiment analysis focus on emotions, as they are the ones that have a target.

Scherer (1984) proposes a different typology of affective states comprised of the following:

- Emotion – a relative short episode of synchronous responses to the evaluation of an internal or external event that is considered of major importance (e.g.: anger, happiness, shame, sadness, enthusiasm etc.)
- Mood – diffuse affective state of low intensity and long duration (e.g.: cheerful, depressive, optimistic, gloomy, apathetic etc.)
- Interpersonal stance – an affective stance taken towards another person in a specific interaction (e.g.: distant, cold, friendly, supportive etc.)
- Attitude – long-lasting, affective coloured belief, preference, predisposition towards objects or people (e.g.: I like, I love, I hate, I want, I appreciate etc.)
- Personality trait – stable personality disposition and typical behavioural tendency (e.g.: anxious, jealous, grumpy, reckless etc.)

According to this typology, Potts (2011) believes that sentiment analysis aims to detect attitudes, since in a text we are dealing with a person having a certain attitude (the source) and a target, a polarity (positive, negative or neutral), and an intensity of that attitude.

This lack of perfect consensus in defining them is only an indicator of the fact that sentiments are mixed and multidimensional. They are context dependent and complex from both social and linguistic perspectives. Identifying them can sometimes be a challenge for humans, let

alone computers. That is why, in order to tackle the sentiment analysis problems, researchers resorted to simplifications. For example, in the context of this problem, Liu (2015) has defined opinion as a quadruple $(\boldsymbol{g}, \boldsymbol{s}, \boldsymbol{h}, \boldsymbol{t})$, where $\boldsymbol{s}$ is the sentiment, $\boldsymbol{g}$ is its target entity, $\boldsymbol{h}$ is the opinion holder and $\boldsymbol{t}$ is the time when the opinion is expressed. The sentiment can have one of the following polarities/orientations: positive, negative or neutral. In order to express its intensity, in practice usually a discrete scale with 5 levels is used (from 1 to 5 or -2 to +2). Alternatively, opinion can also be defined as a qvintuple $(\boldsymbol{e}, \boldsymbol{a}, \boldsymbol{s}, \boldsymbol{h}, \boldsymbol{t})$, where $\boldsymbol{e}$ is the target entity, $\boldsymbol{a}$ is the target aspect of the sentiment, and the rest of the components have the same meaning. In this case, the sentiment analysis is *aspect-based*. An entity $\boldsymbol{e}$ can be represented by a finite set of aspects $\boldsymbol{A} = \{\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots \boldsymbol{a_n}\}$. Also, it can be expressed by any of the entity expressions $\{\boldsymbol{ee_1}, \boldsymbol{ee_2}, \ldots \boldsymbol{ee_s}\}$. In turn, each aspect $\boldsymbol{a}$ of $\boldsymbol{A}$ can be expressed using any of the aspect expressions $\{\boldsymbol{ae_1}, \boldsymbol{ae_2}, \ldots \boldsymbol{ae_m}\}$. Moreover, they can be explicit (e.g.: The picture quality of the camera is great), but also implicit (e.g.: This camera is expensive).

Using these concepts, the researcher proposes the following formalization of the sentiment analysis problem: *Given a document* $\boldsymbol{d}$ *that expresses opinions, find all the qvintuples* $(\boldsymbol{e}, \boldsymbol{a}, \boldsymbol{s}, \boldsymbol{h}, \boldsymbol{t})$ *of* $\boldsymbol{d}$.

2.4. Features

User reviews come in natural language form. Thus, they need to be converted in a form intelligible by a sentiment analysis model. However, reaching such a form is not an easy task. Ideally, we would like to extract from a review as much meaning as possible in order to ease the sentiment decision, but there are several issues that arise during this task:

- *How to treat negation?*

Negation is tricky to deal with, not only because it is sometimes difficult to automatically determine what are the words affected by its presence, but also because its meaning depends based on the intensity of the words it interacts with, not to mention the number of ways in which it can be expressed. For low intensity words such as “good” or “bad”, negating a word is equivalent to

its opposite, but for high intensity ones, such as “superb” or “terrible”, negation has a more general meaning. “Not superb” can basically mean anything from “horrible” to “less superb”.

Correctly marking negation becomes even more important when texts are short, since there are not so many hints of sentiments **(Potts, 2011)**.

- *How to choose the words that are going to be included in the representation of the review?*

Researchers can use all the words in a review in their models, although not all of them are important for the task at hand, or they can select only some of the words, but in doing so they risk losing information. It is a trade-off that needs to be balanced. Usually part-of-speech information is used in making this selection.

One variant is to use only sentiment words (see section 2.4.1).

- *How to represent those features?*

This choice is strongly related to the choice of model. For example, features can be represented as vector spaces, algebraic models based on the idea that important aspects behind the meanings of words are hidden in their distribution i.e. in the occurrence patterns with other words. Such a model is the word-word matrix for a vocabulary V of size n, which is a matrix of size $n \times n$ in which both rows and column represent words in V. The value in cell (i, j) represents the number of times words i and j appear together in the same context, while the row i is a representation of the word i in terms of co-occurrences with other words. When words that have similar meaning are placed near one another in such a space, then we are dealing with a semantic space. Another model is the word-document matrix, which represents documents as distributions of words (Potts, 2011).

Other choices that need to be made when designing a vector space model are the chosen similarity metric or the algorithm for dimensionality reduction. There are many metrics for measuring the similarity between two vectors of size n - A and B. A popular choice when dealing with vectors representing written text is the cosinus distance:

$$cosin(A, B) \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{n}(A_i * B_i)}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2.4.1)$$

Some popular options include:

- feature vector of word frequencies (bag-of-words representations)
- feature vector of word semantic representations

2.4.1. Sentiment words

In such an approach, researchers look in a text for the so-called sentiment words: positive and negative adjectives and adverbs (e.g.: good, wonderful, amazing, terrible, poor, bad etc.) as well as expressions (e.g.: cost an arm and a leg). However, such words are not always enough to make an accurate sentiment analysis. Some of the problems that may appear are the following:

- A word expresses sentiments with different polarities in different contexts

e.g.: *This camera sucks*. versus *This vacuum cleaner really sucks*.

- A sentence contains sentiment words that don't really express a sentiment in the given context

e.g.: *Can you tell me which Sony camera is good?*
If I can find a good camera in the shop, I will buy it.

- Sarcastic sentences

e.g.: *What a great car! It stopped working in two days.*

- A sentence containing no sentiment words can indirectly express the sentiments of the author

ex: *This washer uses a lot of water*. (Liu, 2015)

The general formula of such an approach implies having a lexicon of such sentiment words. It can be an already existing lexicon, or one custom made for the problem at hand. Some existing general lexicons include: Bing Liu Opinion Lexicon (with 2006 positive words, 4783 negative words), SentiWordNet (with scores given to synonym sets in WordNet), Harvard General Inquirer (with semantic and syntactic information attached to words), Linguistic Inquiry and Word Counts, MPQA (Multi-Perspective Question Answering) Subjectivity Cues Lexicon, or ANEW (Affective Norms for English Words). All of these lexicons classify words based on polarity, but their classifications differ as each lexicon has a different starting vocabulary. Thus, researchers sometimes use more than one lexicon and compute an average of their scores (Potts, 2011; Jurafsky, 2018).

The extra effort of creating a custom lexicon for a problem is justified by the fact that general lexicons may not be able to account for norms and sentiments specific to a certain field of study. Before 1990, the proposed sentiment analysis algorithms implied manually building a lexicon of sentiment words and then look for them in the texts to be analysed in order to classify them according to sentiment (Pang & Lee, 2008). After 1990, researchers began using semi-supervised methods when building lexicons starting with some labelled seed sentiment words and patterns (Hatzivassiloglou & McKeown, 1997; Turney, 2002) or some labelled seed sentiment words and synonym/antonym relationships in WordNet (Hu & Liu, 2004; Andreevskaia & Bergler, 2006; Esuli & Sebastiani, 2006; Kim & Hovy, 2006; Godbole, Srinivasaiah, & Skiena, 2007; Rao & Ravichandran, 2009; Blair-Glodensohn, et al., 2008). When using WordNet, at every step of the propagation, the previous words labelled are used as seeds. Thus, the size of the built lexicon grows rapidly.

The next step after deciding on a lexicon is to determine the positivity/subjectivity of a text by applying a function on the sentiment words found in the text.

2.4.2. Bag-of-words representations

The bag-of-words representation makes use of the word-document matrix. This implies creating a vocabulary from the tokens found in all documents and then represent each document as a feature vector that contains for each token in the vocabulary its frequency (word counts).

An alternative is to turn word counts into TF-IDFS (Term Frequency-Inverse Document Frequency). As its name states, TF-IDF is computed as the product of the two statistics: term frequency (the number of times the term appears in a document) and inverse document frequency (how common is the term across all documents). The inverse document frequency is computed as follows:

$$idf(t, D) = \log \frac{N}{|d \in D, t \in d|} \tag{2.4.2}$$

where N is the total number of documents and $|d \in D, t \in d|$ denotes the number of documents in which the term appears.

This transformation addresses the problem of longer documents having higher average count values than shorter documents, even though they might talk about the same topics, and also down weights words that occur in many documents in the corpus and are therefore less informative

than those that occur only in a smaller portion of the corpus, which are two known issues in the word counts features.

However, bag-of-words features lose the information concerning the meaning and position of words. This problem can be addressed by representing documents as vectors of word semantic representations.

2.4.3. Semantic representations

As previously stated, semantic spaces are a way of representing the natural language that wants to incorporate meaning of words by placing words that share context close one to another in the space. Latent Semantic Analysis has such a space as it base. It uses word-paragraph matrices reduced using the Singular Value Decomposition (SVD) technique (Dumais, 2004).

Going one step further, researchers proposed yet another technique for learning word representations, which proved to be superior in conserving linear regularities: feed-forward neural networks (Mikolov, Yih, & Zweig, Linguistic Regularities for Continuous Space Word Representations, 2013; Zhila, Yih, Meek, & Mikolov, 2013). The first model of this kind was proposed by Bengio et al. in 2003. Due to its capacity of representing complex patterns via its hidden non-linear layers, in 2010 Mikolov, Karafiat, Burget, Cernocky, & Khudanpur proposed a recurrent neural network for learning distributed word representations, but they required a very high computational power. To overcome this impediment, in 2013, Mikolov, Chen, Corrado, and Dean develop the word2vec model, a feed-forward neural network with 2 layers. There are 2 variants of architecture for the model: bag-of-words and continuous skip-gram. In the bag-of-words achitecture, the model predicts the current word based on a window of neighbouring words. In skip-gram the order is reversed: the model uses the current word in order to predict the window of neighbouring words, assigning a heavier weight to closer words. While the bag-of-words architecture has the advantage of speed, the skip-gram architecture performs better in the case of less frequent words.

In 2014, Stanford researchers Pennington, Socher, and Manning propose another semantic representation: GloVe, a log-bilinear regression model that learns from the word co-occurence matrix and uses the weighted least squares method in order to minimize the objective function. Moreover, it has the advantage of using the statistics inherent in the matrix.

Ever since their appearance, both word2vec and GloVe have become the norm in machine learning natural language processing models, with researchers using the general pre-trained word representations or training their own. For example, in the SemEval Aspect based Sentiment Analysis 2016 competition, the NPLANG team identified the entity-aspect pairs about which an opinion was expressed (from a set of predefined labels) with an accuracy of 73.031% (F_1 score), as well as the expression through which they were expressed with an accuracy of 72.34%. To do so, they used a 1-layer feed-forward neural network for each aspect and Conditional Random Fields, respectively, which were fed with the following features: all words in the sentence, a list of opinion targets which appear frequently in the training set, the head word of the tree parsing of each sentence for each word, and cluster centres of word representations. All words were represented using word2vec and GloVe models which were trained on Yelp and Amazon reviews. For the first problem, the set of features was completed by the probabilities of belonging to each class as outputted by a convolutional network trained on the continuous representation of words, the list of targets and cluster centres. For the second one, there was an additional recurrent neural network trained on the same features whose output completed the feature set (Toh & Jian, 2016).

2.4.4. Pre-processing

Turning a document into a feature vector most often requires some pre-processing to be done, especially in the case of handcrafted features (i.e. features not learned automatically by a model). Some common pre-processing techniques include: tokenization, stemming, lemmatization, or stopwords and/or punctuation elimination etc.

Tokenization is the process of dividing a text into a set of tokens. A token could be for example a sentence or a word.

Stopwords are words that are very frequent in a language and are thus considered to not contain much information on the analyzed text. Thus, their removal is a common process when dealing with natural language, especially in the information retrieval domain. However, using pronouns, which are considered stopwords, could be a sign of subjectivity and maybe could help predict a higher involvement. Moreover, words like “not”, “no”, “don’t”, which are again considered stopwords, could change entirely the meaning of a sentence. Thus, in the context of sentiment analysis, removing stopwords could strip the text from information and harm the accuracy of the prediction. The research in the field has led to contradictory results regarding this type of preprocessing, with some that support stopwords removal (Bakliwal, et al., 2012; Pak &

Paroubek, 2010; Zhang, Jia, Zhou, & Han, 2012; Speriosu, Sudan, Upadhyay, & Baldridge, 2011; Gokulakrishnan, Pryanthan, Ragavan, Prasath, & Perera, 2012; Kouloumpis, Wilson, & Moore, 2011; Asiaee, Tepper, Banerjee, & Sapiro, 2012) and others that claim stopwords are beneficial to the task (Saif, He, & Alani, 2012; Saif, Fernandez, He, & Alani, 2014; Hu, Tang, Gao, & Liu, 2013; Hu, Tang, & Tang, 2013; Martinez-Camara, Montejo-Raez, Martin-Valdivia, & Urena-Lopez, 2013). Similarly, punctuation marks such as exclamation or question marks could indicate a higher degree of subjectivity and thus, hold sentiment information.

Both stemming and lemmatization have the role of reducing inflected (or sometimes derived) words to their base or root form—generally a written word form. However, in the case of stemming, the root (or stem) need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, in order to reduce the feature set and more easily compare documents, even if this stem is not in itself a valid root. In order to get the valid root, lemmatization should be performed. But lemmatization takes more computational effort, requires the text to be POS-tagged for best results, and most of the time, stemming performs just good enough. One of the most famous and frequently used stemming algorithms is Porter's stemmer.

2.5. Supervised learning

The sentiment analysis problem was, and still is, frequently turned into a classification problem that is solved by building a model and exposing it to documents labelled with sentiment classes (e.g.: positive/negative) so that it could learn to discern between those classes. Such an approach falls under the supervised learning paradigm, which states the following:

Given a set of N independent and identically distributed examples of the form (input, output)$(x_1, y_1), (x_2, y_2)$ ... (x_N, y_N), where each y_j was generated using an unknown function $y = f(x)$, find the function h that best approximates the real function f. When the output y belongs to a finite set of values, the learning problem is called *classification*. Otherwise, it is called *regression* (Norvig & Russel, 2010).

The h function is called *hypothesis*. We say a hypothesis generalizes well if it can predict the right output for new and unseen examples. Its accuracy is measured on a *test set,* composed of these new examples*,* which is different from the *train set,* that contains the examples the model was trained on.

In general, there is a compromise between complex hypotheses that perform well on the train set and simpler hypotheses that can generalize better. Choosing a hypothesis that is too complex can lead to the phenomenon of *overfitting*: high performance on the train set but low performance on the test set (Duda, Hart, & Stork, 2000). The hypothesis selection is done in two phases: first, choosing the hypothsesis space, followed by choosing the best hypothesis in that space. When choosing the hypothesis space, the need for yet another set arises: a *validation set.* On this set, the error decreases as the complexity grows until it reaches a minimum (which corresponds to the desired hypothesis). After that, the overfitting phenomenon appears. (Figure 1) (Norvig & Russel, 2010).

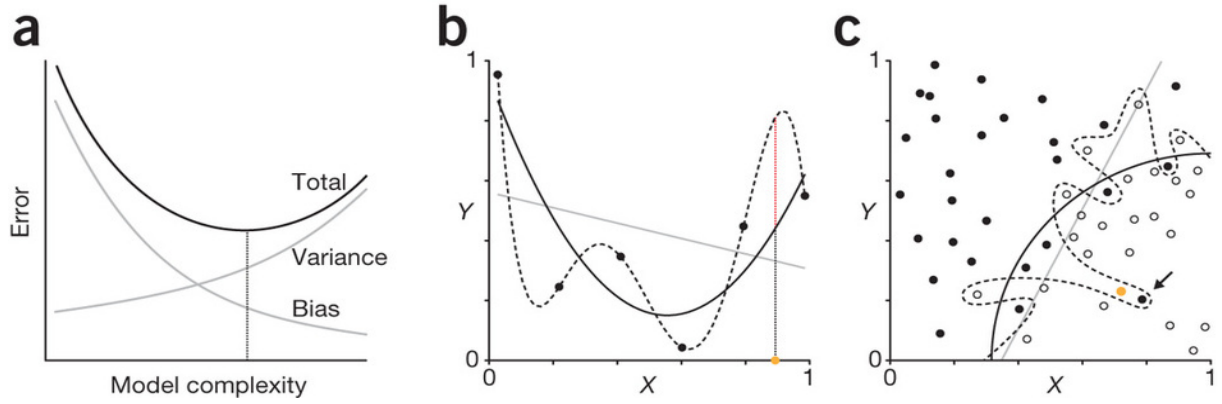

Figure 1. Model selection. (Lever, Krzywinski, & Altman, 2016)

(a) When complexity grows, the variance increases, while the bias decreases; (b) the grey line corresponds to underfitting, the black one shows a reasonable approcimation of the points, while the dotted one corresponds to overfitting; (c) the grey line corresponds to underfitting, the black one shows a reasonable approcimation of the points, while the dotted one corresponds to overfitting

The performance of a learning function is measured using the so-called loss function, defined as the utility that is lost when estimating $h(x) = \hat{y}$, while the right answer was $f(x) = y$. Thus, the agent aims to minimize the generalization loss:

$$GenLoss_L(h) = \sum_{(x,y)\in\mathcal{E}} L\big(y, h(x)\big)P(x,y) \tag{2.5.1}$$

by choosing the best hypothesis

$$h^* = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\, GenLoss_L(h),\ \mathfrak{H} = \text{hypotheses space} \tag{2.5.2}$$

where L is the loss function, ε is the set of all possible examples, and $P(X,Y)$ is the probability distribution of the examples.

Since $P(x, y)$ is not known, the learning agent can only estimate the generalization loss with the empirical loss on a set of examples E:

$$EmpLoss_{L,E}(h) = \frac{1}{N}\sum_{(x,y)\in E} L\big(y, h(x)\big) \tag{2.5.3}$$

and try to minimize it by choosing the hypothesis:

$$h^* = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\, EmpLoss_{L,E}(h) \tag{2.5.4}$$

Pang, Lee and Vaithyanathan (2002) were among the first to employ a supervised, machine learning approach to this sentiment analysis problem. Given the promising results of Naïve Bayes, Maximum Entropy and Support Vector Machines (SVM) on topic-based categorization, they experimented with these models, and found SVM to perform better than the other two classifiers, but overall the registered results were not comparable with the state-of-the-art in that field, due to the particularities of the sentiment discourse. Nevertheless, these models still remain the most common when creating a baseline for a sentiment analysis solution. In fact, they can still obtain comparable results to the ones outputted by more complex models, if fed with the right features. For example, in the SemEval Aspect Based Sentiment Analysis 2016, one of the winning teams obtained an accuracy of 81.93% on the recognition of the dominant sentiment with respect to the identified aspects at the sentence level with a Maximum Entropy classifier trained on bag-of-words representations (Hercig, Brychcin, Svodoba, & Konkol, 2016), while another identified the aspects with an accuracy of 83.995% using a SVM fed with the nouns, adjectives, adverbs in the text, their lemmas, their part-of-speech (POS) tags and all the bigrams (Alvarez-Lopez, Juncal-Martinez, Fernandez-Gavilanes, Costa-Montenegro, & Gonzalez-Castaño, 2016).

However, the best results so far have been obtained with neural networks in the form of recurrent or convolutional networks. The best accuracy up to this point on fine-grained classification – 70.02% was reported by Howard and Ruder (2018), who proposed ULMFiT, an universal transfer learning technique, that improved the results of a Long Short Term Memory (LSTM) classifier. This technique is composed of three stages. First, a language model is trained on a general-domain corpus. Then, the full model is fine-tuned on target task data using discriminative fine-tuning and slanted triangular learning rates so as to learn the specific patterns. Finally, the classifier is fine-tuned on the target task data using gradual unfreezing and the same discriminative fine-tuning and slanted triangular learning rates techniques to preserve low-level representations and adapt high-level ones. The language model they use when reaching the

reported high accuracy is the AWD-LSTM. The second best result (69.42% accuracy) is obtained by Johnson and Zhang (2017), who used a Deep Pyramid Convolutional Neural Network (DPCNN), an architecture that is able to capture long-range associations in texts. Good accuracy (67.61%) was reported by the same authors the previous year with yet another convolutional network architecture: a shallow (word-level) convolutional network taking in two-views embeddings. A year prior to that, the best results were obtained, on the contrary, by a character-level convolutional network (Zhang, Zhao, & LeCun, 2015). Other convolutional network approaches were proposed by Dos Santos and Gatti (2014), Kalchbrenner, Grefenstette, and Blunsom (2014), and Lai , Xu, Liu, & Zhao (2015).

2.5.1. Classifier evaluation

The most frequent used method of evaluating a classifier is the accuracy, defined as:

$$accuracy = \frac{number\ of\ right\ predictions}{total\ number\ of\ predictions} \tag{2.5.18}$$

However, this method alone does fail to depict a real image of the performance when dealing with unbalanced classes, this being a frequent problem in the field of natural language processing. Thus, in order to get a more complete image of the results, together with accuracy, the following measures are also computed.

$$precision\ = \frac{True\ Positives}{True\ Positives+False\ Positives} \tag{2.5.19}$$

$$recall = \frac{True\ Positives}{True\ Positives+False\ Negatives} \tag{2.5.20}$$

These two measures can be combined in the F1-score.

$$F_1 = 2 * \frac{precision*recall}{precision+recall} \in [0,1] \tag{2.5.21}$$

If the F1-score is close to 1, then the accuracy of the classification is high.

2.6. Unsupervised learning

Unsupervised learning is a type of learning in which the agent learns without having an implicit feedback. One example of unsupervised learning task is topic modelling (i.e. building a

statistical model for discovering the abstract topics hidden in a collection of documents). The most common model employed for this task is Latent Dirichlet Allocation (LDA).

2.6.1. Latent Dirichlet Allocation

Proposed in 2003 by Blei, Ng and Jordan, LDA is a generative probabilistic model – a three-level Bayesian model – that models a document as a finite distribution of topics and a topic as a distribution of words. It is an unsupervised learning approach that aims to discover the grouping of words into topics, starting from given initial distributions and number of topics. It does so using approximate inference techniques based on variational methods and an Expectation Maximization (EM) algorithm for empirical Bayes parameter estimation.

In order to validate the model and measure coherence of topics, the authors propose a metric called perplexity, defined as the inverse of the geometric mean per-word likelihood. Thus, the lower the perplexity, the higher the likelihood on the test data.

$$perplexity(D_{test}) = e^{-\frac{\sum_{d=1}^{M} logp(\boldsymbol{w}_d)}{\sum_{d=1}^{M} N_d}} \tag{2.6.1}$$

where $\boldsymbol{w}_d$ is the word vector corresponding to document d

N_d is the size of document d.

The studies of Mimno, Wallach, Talley, Leenders, & McCallum, 2011 have shown, however, that perplexity is not so well correlated with human judgement. Thus, they propose the UMass metric. This metric uses a pairwise score function:

$$score(w_i, w_j) = log\frac{D(w_i,w_j)+1}{D(w_i)} \tag{2.6.2}$$

which is the empirical conditional log-probability $logp(w_j|w_i) = log\frac{p(w_i,w_j)}{p(w_j)}$ smoothed by adding 1 to the numerator. $D(w_i)$ is the number of documents containing the word w_i, while $D(w_i, w_j)$ is the number of documents containing both words. UMass measures how much, within the words used to describe a topic, a more frequent term is in average a good predictor for a less frequent one.

3. Dataset

In order to apply supervised learning, an annotated dataset of restaurant reviews was needed. One such dataset was made available by Yelp at https://www.yelp.com/dataset in the form of the Yelp Open Dataset – "an all-purpose dataset for learning". The dataset, available in both JSON and SQL files, is a subset of Yelp's businesses, reviews, and user data for use in personal, educational, and academic purposes. It contains 5,200,000 reviews on 174,000 businesses from 11 metropolitan areas, together with over 1,200,000 business attributes like hours, parking, availability, ambience, aggregated check-ins over time for each business, 1,100,000 tips by 1,300,000 users and 200,000 pictures.

The JSON version of the dataset comes in the form of 6 files: business.json, checkin.json, review.json, tip.json, user.json, photos.json. A preview of their content can be visualized in Appendix 1. Out of the 5,200,000 reviews, only the ones related to businesses in the food industry (restaurants) were selected – thus, reviews that had a business_id associated with a categories array that contained "Restaurants". In the end, the corpus consisted of 3,221,419 reviews, out of which 362,143 (11.24%) were 1 star reviews, 310,257 (9.63%) were 2 stars reviews, 451,004 (14%) were 3 stars reviews, 883,531(27.42%) were 4 stars reviews and 1,214,484 (37.7%) were 5 stars reviews.

4. Features

The text analysis process was an incremental one, gradually increasing complexity, in the hope of carefully finding the recipe that offered the best accuracy for the problem at hand. Thus, the feature extraction process followed the same path, starting with some simpler features being extracted and then getting down to more complex ones.

4.1. Bag of words representations

At first, the review content was treated in a **bag of words** manner. This implied creating a vocabulary from the tokens found in all documents (reviews) and then representing each document (review) as a feature vector that contained for each token in the vocabulary its frequency. In order to observe the effect of different types of features on the classifier accuracy, a trial and error approach was followed, and many vocabularies were created, depending of the type of pre-processing made on the text. More precisely, the observations were centred around the effects of the presence/absence of punctuation, stopwords and stemming on the performance of the classifier. As a result, different versions of pre-processing were proposed, each resulting in a different set of features:

Table 1. Feature sets and the pre-processing pipelines associated

	Sentence tokenization	Word tokenization	Lowercasing	Stopwords removal	Punctuation removal	Stemming*
Feature set 1	✓	✓	✓	✓	✓	
Feature set 2	✓	✓	✓	✓	✓	✓
Feature set 3	✓	✓	✓		✓	
Feature set 4	✓	✓	✓	✓		

Feature set 5	✓	✓	✓			
Feature set 6	✓	✓	✓			✓

* Stemming was performed using one of the most famous and frequently used algorithms: Porter's stemmer.

In order to refine the features, the word counts generated before were turned into TF-IDFS (Term Frequency-Inverse Document Frequency.

Both sets of features, in their different versions depending on the preprocessing function applied, were saved as sparse matrices.

4.2. GloVe word embeddings

Word embeddings provide us with a way of representing words in a computer-intelligible way that is still able to preserve some meaning through the concept of distance. Two words that are close in this continuous space determined by the embeddings are also similar in concept. The GloVe word vectors, pre-trained on a Wikipedia corpus (https://nlp.stanford.edu/projects/glove/), were chosen as continuous representation for the words in the dataset. Thus, each document became a list of word embeddings of size (1234, 50), where 1234 is the maximum number of tokens (words) in a document and 50 is the dimension of a word representation. For reviews that contained less words, padding was applied, so that all lists have the same size. This was necessary since these features were going to be fed to a neural network.

4.3. Restaurant sentiment embeddings

On one hand, a pre-trained model offers a general representation of words. On the other, it is known that words change meaning when used in different contexts. Thus, it is only natural to look for a representation that could capture the word relationships specific to this field. As a result, a custom "restaurant sentiment embeddings" was built. For that, I turned to transfer learning, chose the previous general GloVe word embeddings as starting point and then fine-tuned them on the restaurant dataset. To do so, the embedding layer of the proposed convolutional networks was

allowed to update its weights in response to the classification feedback. In this way, word representations can adjust to the meaning of words in this specific restaurant review context.

4.4. Handcrafted features

Assuming that the topics presented in a review have also an impact on the final review score, the review representation as topic distribution was included in the set of input features, which is a novel approach. Moreover, for the most prominent topics in a review, a sentiment score was computed based on the positive and negative children in the dependency graph of its most important words.

4.4.1. LDA model

But first, in order to get the topic distribution, an unsupervised learning approach in the form of Latent Dirichlet Allocation (LDA) was adopted. The corpus for training this model was obtained from the Yelp!'s restaurant reviews, by applying the following pre-processing pipeline:

1. Tokenize the documents
2. Keep only nouns and verbs (as they are most representative for a topic)
3. Lowercase the words
4. Remove punctuation
5. Apply lemmatization
6. Remove least frequent 5% words
7. Remove most frequent 15% words
8. Represent each document as an array of word frequencies

From the corpus, 90% of reviews were actually used for fitting the model, as train set, while 10% were employed in evaluating the model, as test set. As a training strategy, online learning was used, due to its speed advantages. This means that at each EM update, a mini-batch of data was used to update the parameters. The learning rate was decaying by a factor of 0.5. Also, early iterations were downweighed by 10.

Due to the fact that the number of topics is an input variable, I used cross-validation to discover the best parameter from the dataset. At first, perplexity was chosen as evaluation metric for the models, but this measure would always decrease with the number of topics, implying that increasing the number of topics would always lead to a better model. This approach, however, was not feasible in terms of human interpretability. That is why the evaluation of topic models needed,

next to holdout perplexity, an additional measure in the form of the UMass that can rate topics with respect to human understandability. This measure was employed in a four-stage pipeline proposed by Michael, Andreas, & Alexander (2015). First, the word set is segmented into a set of pairs of word subsets. Next, word probabilities are computed based on a given reference corpus. Both, the set of word subsets as well as the computed probabilities are consumed by the confirmation measure to calculate the agreements of pairs (how much one pair supports the other). Last, those values are aggregated to a single coherence value, which in my case was given by the UMass metric.

The coherence of the LDA models trained on the entire corpus of reviews with different number of topics can be seen in Figure 2.

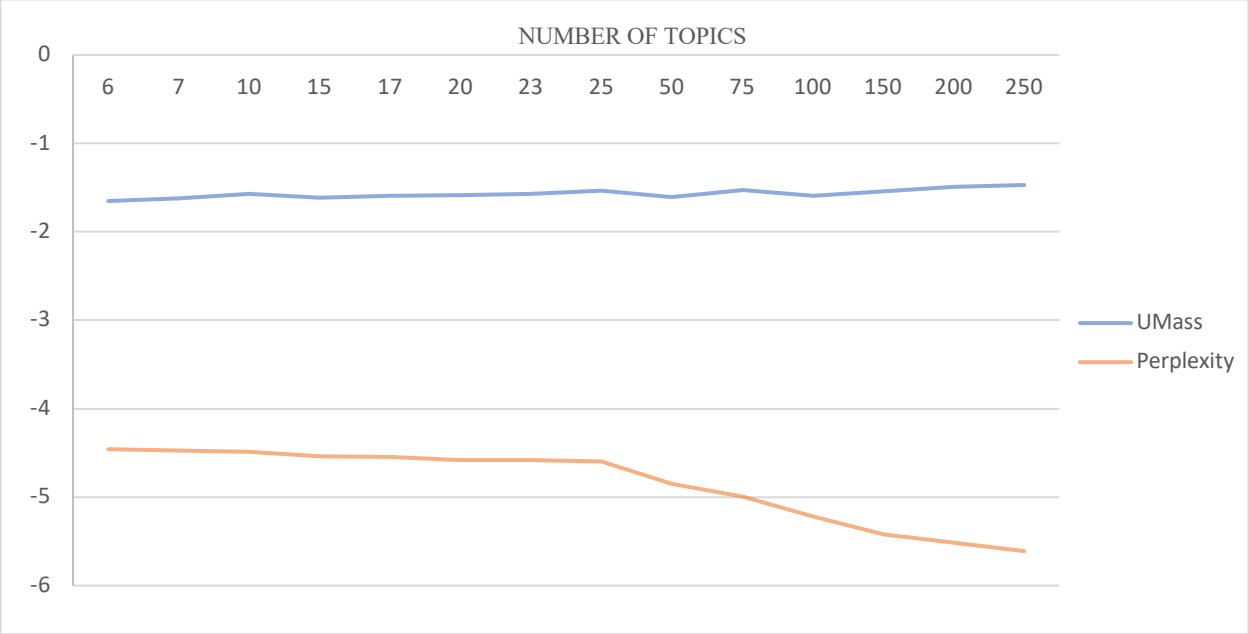

Figure 2. LDA model coherence

As it can be seen, the UMass metric outputted similar results for different numbers of topics (the differences are up to 0.2). Moreover, as it is only natural, given the fact that I was working with numerical approximations and random sampling, the results were different at every run of the algorithm, and these differences were of the same magnitude. For example, for the 75 topics model, I obtained a coherence measure of -1.52, -1.57, -1.60 at different runs. Thus, I proceeded to manually inspecting some of the models in order to choose the one best outputting the most distinctive and coherent topics for the restaurant field.

6-topic model

The first model examined was the 6-topic model. According to it, the most relevant terms for each topic are the ones presented in Table 2 in order of their probability in the multinomial topic distribution. By making sense of these words as group, I tried to give each topic a human understandable name. The naming exercise together with the topic visualization offered by pyLDAvis (Figure 3) indicated that, while the topics are quite distinctive one from the other, they are still quite general and could be further divided into more granular sub-topics. For example, topic no. 2 could refer to various aspect such as location, bar or price.

Table 2. Topics and their most relevant words for the 6-topic model

Topic	Relevant words	Aspect
Topic 1	order, get, food, go, come, time, wait, say, take, table, ask, service, make, place, give, tell, want, server, minutes, drink, know, leave, seat, eat, people, experience, waitress, sit, restaurant, think	**Waiting services**
Topic 2	food, place, go, service, bar, make, time, get, drink, price, area, menu, great, restaurant, staff, find, visit, spot, quality, location, night, look, enjoy, experience, lot, come, try, vegas, feel, dinner	**??**
Topic 3	food, place, love, service, go, recommend, time, try, get, amaze, come, eat, star, price, staff, give, everything, vegas, say, want, friends, like, people, know, thing, look, need, day, bring, way	**Recommendable**
Topic 4	chicken, salad, sandwich, get, meat, soup, bread, beef, order, lunch, go, place, come, try, eat, side, sauce, time, make, say, think, taste, bite, meal, like, take, want, menu, look, serve	**Lunch**
Topic 5	dish, taste, restaurant, order, sauce, flavour, rice, roll, portion, menu, try, come, bite, serve, cook, meal, make, price, side, think, like, eat, enjoy, dinner, everything, look, lot, use, recommend, expect	**Fine dining**
Topic 6	fry, pizza, burger, cheese, order, sauce, get, taste, try, make, cook, bite, flavour, eat, think, side, like, home, way, use, want, nothing, come, thing, look, lot, place, expect, decide, something	**Fast food**

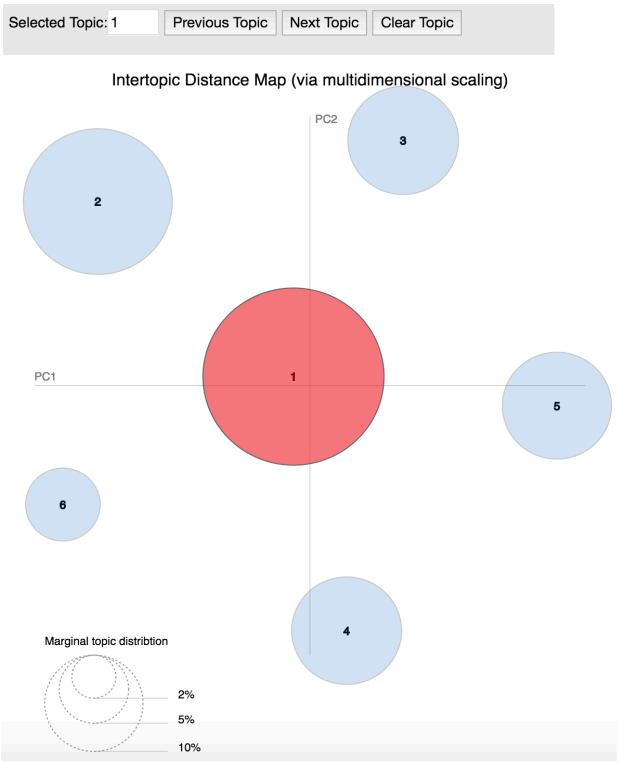

Figure 3. Top 30 Most Relevant Terms for Topic 1 (pyLDAvis)

17-topic model

Going from 6 to 17 topics, we can see an increase in the granularity of each topic (Table 2). However, there are still topics that do not have yet a coherent human understandable meaning (Topic 11) or a single point of focus (Topics 15 and 16), or even topics that appear to refer to the same aspect of the restaurant experience (Topics 15 and 16). The topics outputted by the model can be visualized in Figure 4. We can see that they start to get closer in the map, as they become more fine-grained.

Table 3. Topics and their most relevant words for the 17-topic model

Topic	Topic coherence (UMass)	Relevant words	Aspect
Topic 1	-1.288	time, food, go, dinner, get, service, order, make, wait, restaurant,	**Time**

Topic 2	**-1.025**	roll, come, order, place, go, get, restaurant, try, say , food	**Roll**
Topic 3	-1.300	love, get, place, food, service, go, drink, time, wait, take	**Customer satisfaction**
Topic 4	-1.728	food, price, service, great, lunch, place, star, go, co me, spot	**Price**
Topic 5	-1.363	place, go, drink, get, make, food, eat, come, bar, think	**Drinks**
Topic 6	-1.145	get, order, come, wait, go, take, time, ask, table, food	**Waiting services**
Topic 7	-1.392	meat, bread, cheese, get, serve, come, go, taste, restaurant, side	**Food**
Topic 8	-1.265	rice, order, beef, soup, salad, eat, place, food, get, dish	**Dishes**
Topic 9	-1.096	chicken, order, fry, food, get, make, come, place, go, sauce	**Chicken**
Topic 10	-1.170	try, place, look, order, go, menu, food, come, like, time	**Experimenting**
Topic 11	-1.486	time, place, bar, food, visit, go, service, staff, use, get	**??**
Topic 12	-1.392	place, sauce, say, get, order, go, food, want, know, cook	**Sauce**
Topic 13	-1.383	sandwich, get, place, make, go, come, food, everything, try, lunch	**Sandwich**
Topic 14	-1.490	pizza, place, order, go, make, food, get, salad, eat	**Pizza**
Topic 15	**-1.738**	food, restaurant, staff, service, place, eat, seat, recommend, take, lot	**Food and service**
Topic 16	-1.160	dish, service, food, menu, place, come, table, order, try, make	**Food and service**
Topic 17	-1.127	burger, fry, go, location, food, get, order, come, time, say	**Burger**

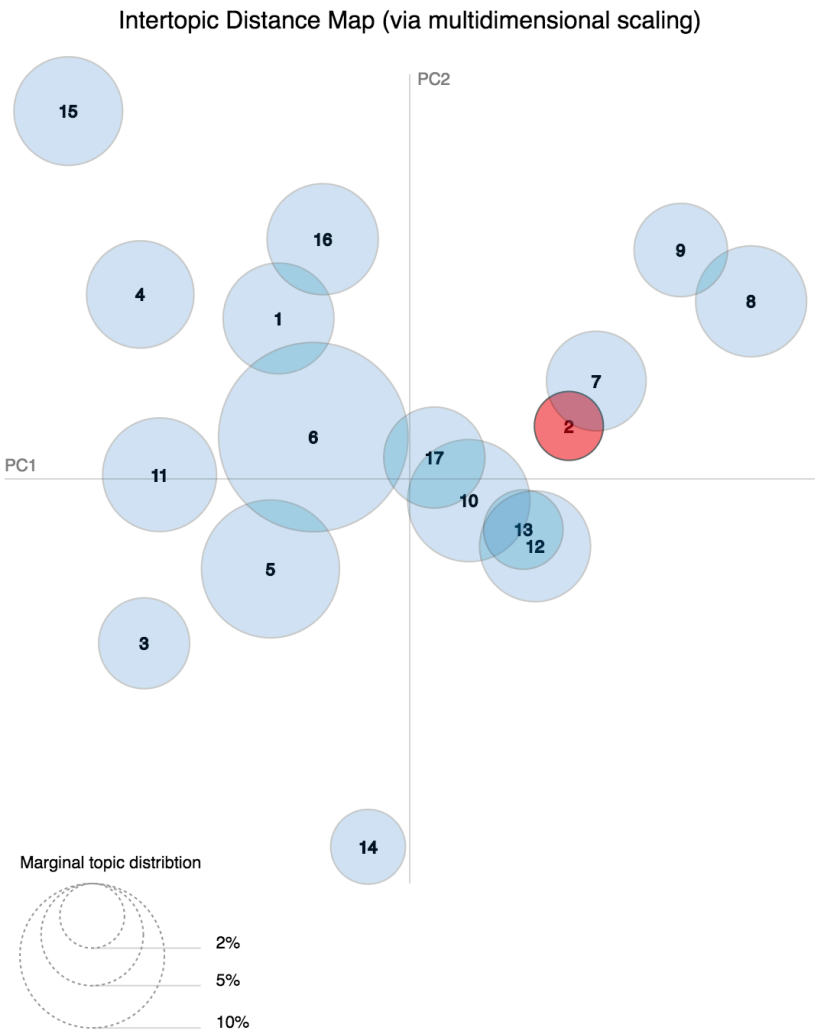

Figure 4. 17-topic model. Intertopic distance map (pyLDAvis)

20-topic model

Further increasing the number of topics, more coherent, single focused topics, that correspond to more subtle aspects (waiting services, waiting time, seating, sauce, salad) are obtained. Of course, there are still topics that do not make much sense (ex: Topic 19). Also, from the visualization (Figure 5) it can be observed that topics get closer or even intertwined (e.g.: topic 10 and 15). This enables us to make affirmations such as:

"People that talk about rice, soup and meat also talk about sauce."

"People that talk about waiting time, also talk about drinks and bar."

"People that talk about lunch also talk about location."

Such correlations could be of interest to market researchers.

Table 4. Topics and their most relevant words for the 20-topic model

Topic	Topic coherence (UMass)	Relevant words	Aspect
Topic 1	-1.151	chicken, order, food, fry, get, come, go, rice, place, time	**Chicken**
Topic 2	-1.568	try, taste, place, food, flavour, everything, come, order, give, nothing	**Experimenting**
Topic 3	-1.570	dish, salad, beef, order, come, menu, meal, enjoy, side, restaurant	**Salad**
Topic 4	-1.374	sauce, cheese, side, order, get, go, flavor, meat, bi te, come	**Sauce**
Topic 5	-1.304	love, location, place, time, stop, food, eat, go, stuff, get	**Location**
Topic 6	-1.538	table, restaurant, experience, food, seat, service, go, night, seat, dinner	**Seating**
Topic 7	-1.254	rice, soup, meat, order, beef, dish, come, taste, eat, get	**Rice, soup, meat**
Topic 8	**-0.929**	ask, server, check, come, order, bring, service, time, get, food	**Billing services**
Topic 9	-1.437	place, menu, look, find, see, offer, go, area, food, make	**Menu/Offering**
Topic 10	-1.346	make, Vegas, feel, visit, time, eat, food, go, place, home	**Vegas**
Topic 11	-1.215	say, order, tell, waitress, come, go, get, ask, take, want	**Waiting services**
Topic 12	-0.939	wait, time, order, food, minutes, take, come, get, service, go	**Waiting time**

Topic 13	-1.401	fry, burger, order, get, place, go, cheese, come, eat, cook	**Burgers**
Topic 14	**-1.917**	food, service, place, great, amaze, go, spot, love, price, recommend	**Recommendable**
Topic 15	-1.236	drink, bar, night, get, sit, go, food, come, service, area	**Bar**
Topic 16	-1.528	food, lunch, price, place, portion, recommend, come, staff, service, eat	**Lunch**
Topic 17	-1.530	sandwich, bread, cheese, get, make, go, lunch, meat, try, bite	**Sandwiches**
	-1.146	pizza, roll, order, place, get, go, try, make, eat, come	**Pizza**
Topic 19	-1.422	get, place, go, time, think, food, price, come, pay, eat	**??**
Topic 20	-1.195	star, give, dinner, food, review, cook, service, eat, get, go	**Review**

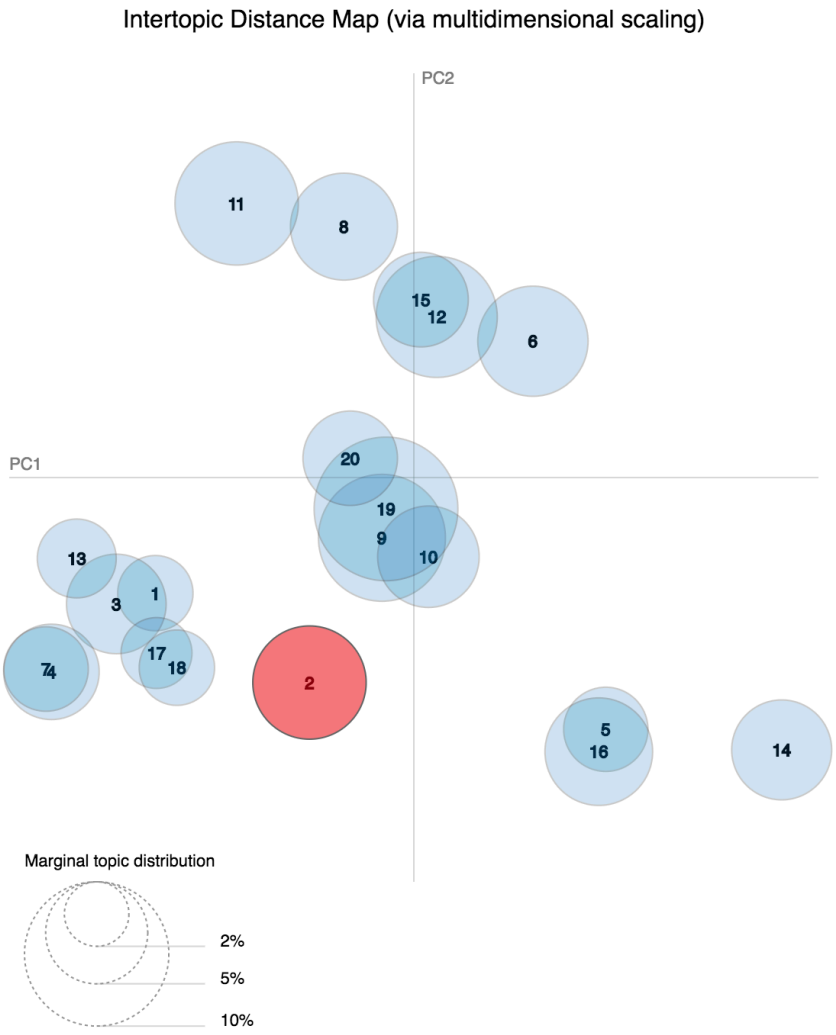

Figure 5. 20-topic model. Intertopic distance map (pyLDAvis)

25-topic model

If we were to look at the intertopic distance map (Figure 6), the 25-topic model brings less overlap than the previous model, as well as more granularity. Interesting to note is the fact that the topics that are similar from human perspective are also similar from a mathematical point of view and represented close one to another. Also, the topics that overlap are very much coherent in the associations that they lead to:

"People that talk about sandwich taste also talk about sauce."

"People that talk about burgers also talk about cheese."

Table 5. Topics and their most relevant words for the 25-topic model

Topic	Topic coherence (UMass)	Relevant words	Aspect
Topic 1	-1.205	time, come, food, service, place, go, drink, eat order, say	**??**
Topic 2	-1.309	Vegas, food, quality, restaurant, place, try, enjoy, me nu, service, go	**Quality**
Topic 3	-1.019	restaurant, location, time, service, food, visit, go, order, place, get	**Location**
Topic 4	-1.280	roll, go, place, need, get, order, time, service, try, wa y	**Rolls**
Topic 5	-1.304	chicken, rice, order, get, come, food, taste, place, try, soup	**Chicken**
Topic 6	-1.297	pizza, drink, place, go, get, night, order, make, try, think	**Pizza**
Topic 7	-1.222	say, ask, order, go, tell, come, food, get, want, bring	**Waiting services**
Topic 8	-1.382	dish, flavour, food, try, taste, restaurant, place, think, service, cook	**Dish flavour**
Topic 9	-1.253	salad, beef, go, meal, time, food, meat, place, get, taste	**Dishes**
Topic 10	-1.457	fry, food, pay, go, place, service, nothing, get, order, check	**Fast food price**
Topic 11	-1.823	get, taste, sandwich, bread, star, flavour, cheese, give, make, meat	**Sandwich taste**
Topic 12	1.622	bar, area, menu, place, try, go, look, walk, see, sit	**Bar**
Topic 13	-1.182	meat, food, get, use, place, order, go, chicken, service, make	**Meat**
Topic 14	**-0.841**	wait, minutes, go, order, get, take, time, food, service, come	**Waiting time**

Topic 15	-1.232	place, sandwich, price, recommend, try, go, get, find, food, time	**Sandwich price**
Topic 16	-1.076	get, order, place, people, go, time, say, make, take, food	**??**
Topic 17	-1.489	food, price, menu, soup, portion, service, place, give, like, order	**Price**
Topic 18	-1.085	eat, come, bite, side, food, think, get, look, place, order	**Eating**
Topic 19	-1.180	table, get, seat, server, food, come, wait, place, sit, service	**Seating**
Topic 20	-1.506	place, food, staff, go, love, get, service, eat, lot, price	**Staff**
Topic 21	**-1.893**	love, great, service, food, spot, lunch, dinner, enjoy, stop, place	**Customer satisfaction**
Topic 22	-1.491	sauce, order, serve, come, dish, try, menu, make, meal, side	**Sauce**
Topic 23	-1.190	want, cheese, get, order, time, go, place, make, lunch, try	**Cheese**
Topic 24	-1.169	food, make, experience, service, take, order, restaurant, give, time, get	**Experience**
Topic 25	-1.285	burger, fry, get, order, time, try, night, come, food, bar	**Burgers**

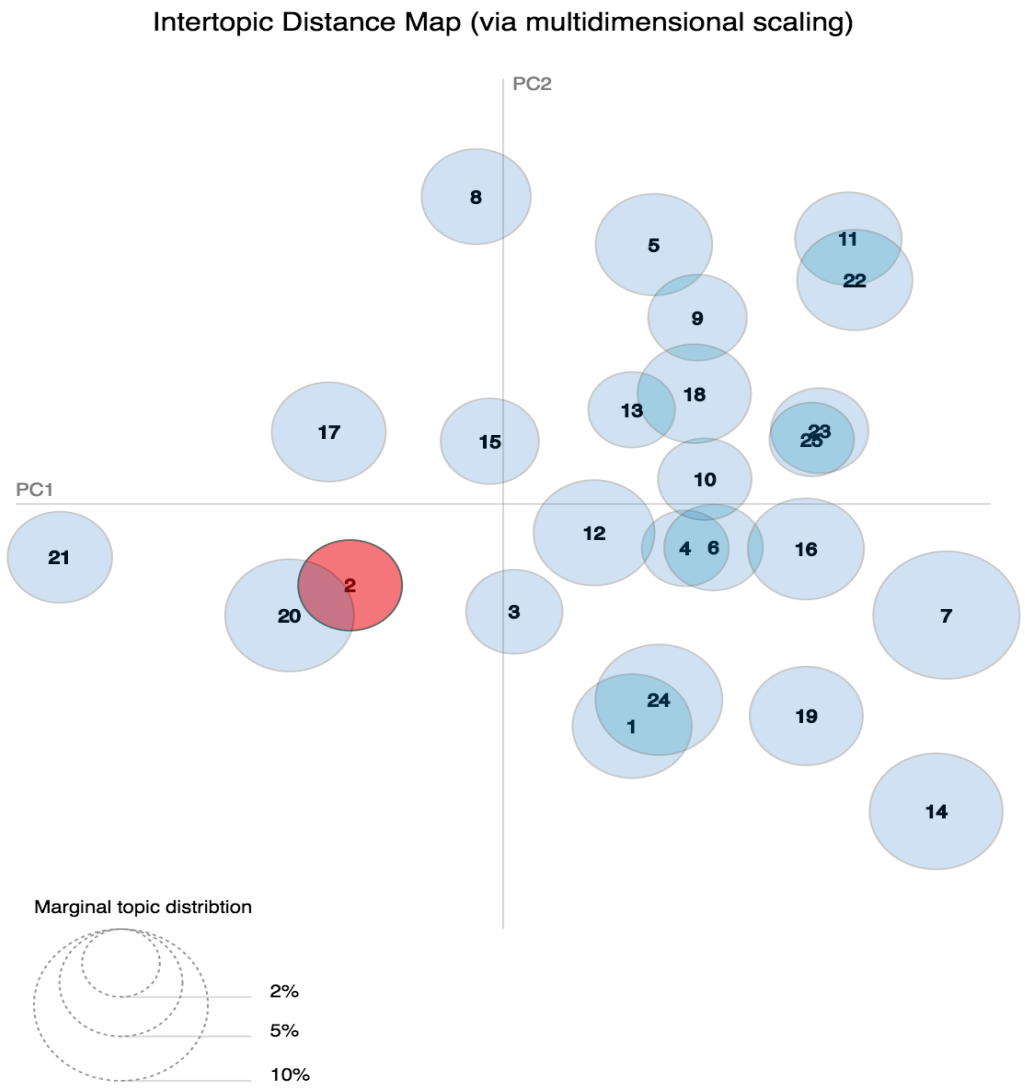

Figure 6. 25-topic model. Intertopic distance map (pyLDAvis)

30-topic model

The model with 30 topics, however, did not impress from a human understandability point of view. Although it captured some interesting aspects such as cooking time or review appearance, it had topics without any real meaning (topics 16 and 17) and also topics determined by only one dominant word, which in some cases, was not very meaningful (go, place). Moreover, there were still topics without a single point of focus (again the "food and service" general topic). Looking at the visualization (Figure 7), we see that the topics determined are even more similar one to another,

as they are more fine-grained. However, there seems to be a bit too much overlap and some over-clustering that gets to clusters formed of a single word with high probability.

Table 6. Topics and their most relevant words for the 30-topic model

Topic	Topic coherence (UMass)	Relevant words	Aspect
Topic 1	**-0.833**	wait, order, minutes, food, time, take, come	**Waiting time**
Topic 2	-1.173	cheese, bread, try, time, go, get, come, think, order, bite	**Cheese, bread**
Topic 3	-1.156	meat, get, order, side, time, bite, eat, sauce, place, go	**Meat dishes**
Topic 4	-1.190	burger, fry, get, go, place, try, order, come, want, taste	**Burgers**
Topic 5	-0.972	menu, restaurant, go, time, order, look, make, place, get, come	**Menu**
Topic 6	-1.069	star, place, look, get, go, live, food, make, order, say	**Review look**
Topic 7	-1.075	eat, food, get, tell, service, time, place, take, make, go	**Eating**
Topic 8	-1.122	chicken, order, food, rice, fry, time, take, come, go	**Chicken**
Topic 9	-1.130	love, place, try, get, time, pizza, come, pizza, go, food, order	**Customer satisfaction**
Topic 10	-1.193	say, ask, order, want, think, go, get, place, get, tell, know	**Knowledge**
Topic 11	-1.196	salad, come, order, side, make, food, try, service, place, everything	**Salad**
Topic 12	-1.210	give, food, star, get, order, portion, try, place, go, think	**Review food**
Topic 13	-1.234	taste, dish, food, flavour, place, try, menu, restaurant, eat, order	**Dish taste**
Topic 14	-1.240	order, come, get, go, dish, place, cook, time, food, taste	**Cooking time**

Topic 15	-1.285	restaurant, sauce, place, friends, food, serve, go, recommend, service, get	**??**
Topic 16	-1.300	table, make, work, service, order, look, go, sit, seat, food	**Seating**
Topic 17	-1.302	food, think, place, try, get, come, like, go, time, lot	**??**
Topic 18	-1.310	sandwich, order, get, food, come, go, time, make, try, bread	**Sandwich**
Topic 19	-1.318	bar, get, food, place, wait, sit, go, time, area, service	**Bar**
Topic 20	-1.328	roll, lot, try, go, dish, place, get, rice, order, price	**Rolls**
Topic 21	-1.341	come, server, dinner, table, order, soup, restaurant, food, meal, service	**Table service**
Topic 22	-1.406	pizza, order, food, go, get, service, fry, place, price, sauce	**Pizza**
Topic 23	-1.411	drink, get, come, go, people, night, food, place, bar, sit	**Drinks**
Topic 24	-1.456	go, food, time, location, service, get, place, love, want, staff	**Go?**
Topic 25	-1.512	place, Vegas, price, find, try, food, quality, come, visit	**Vegas**
Topic 26	-1.533	experience, food, nothing, service, make, feel, time, know, place, day	**Experience**
Topic 27	-1.562	place, make, staff, try, get, beef, go, recommend, use, food	**Place?**
Topic 28	-1.574	seem, place, check, get, time, bring, take, come, see, walk	**Appearance**
Topic 29	-1.577	food, service, great, place, price, time, staff, come, amaze, get	**Food and service**
Topic 30	**-1.600**	lunch, enjoy, come, place, stop, food, get, area, order, spot	**Lunch**

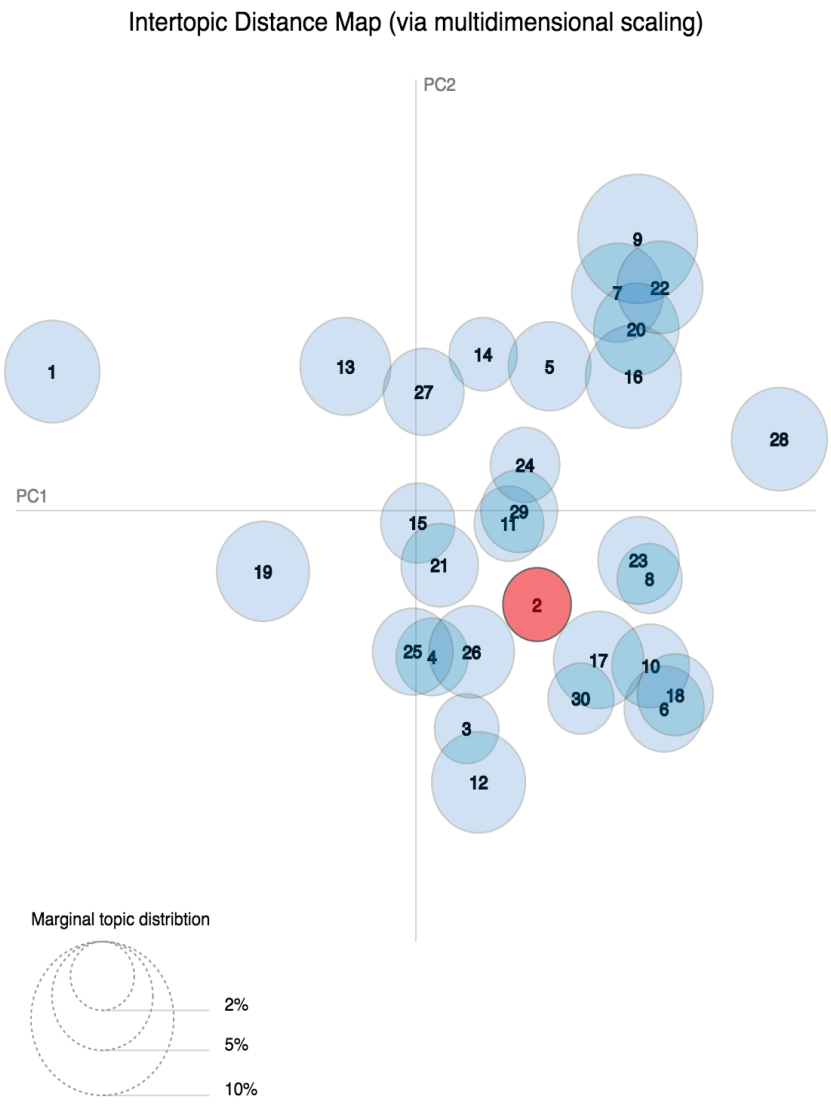

Figure 7. 30-topic model. Intertopic distance map (pyLDAvis)

4.4.2. Handcrafted topic features

Although the coherence of a topic grew as the number of topics increased, it was considered that the 25-topic model contains just enough granularity and diversity of topics as well as correlations so as to offer insights to researchers. Thus, it was further used to model reviews as topic distributions, leading to the so-called handcrafted topic features (i.e. an array of 25 real elements containing the probabilities of belonging to each of the 25 discovered topics).

4.4.3. Handcrafted sentiment topic features

In order to compute the sentiment score per topic of a review, a lexicon-based approach was adopted. More precisely, the 10 most important words for a topic were chosen (the ones that had the highest probability in the topic distribution) and searched for in the review. If a word was found, its dependency graph was computed. If among the children of the head of the dependency graph, a sentiment word was found, the sentiment score would be updated. Generally speaking, a sentiment word is a word that expresses a sentiment regarding a target. For my experiments, I narrowed this definition to adjectives, adverbs, nouns or verbs from the ANEW or Bing Liu Opinion Lexicons. Thus, in order to determine if a word was a sentiment word I would check its part of speech and look for it or its lemma in these two lexicons.

The sentiment score had 3 dimensions: anew, positive and negative. The anew dimension was computed as the sum of the ANEW scores of the sentiment words found in the review as described previously, which were also found in the ANEW lexicon. The ANEW lexicon rates words in terms of pleasure/valence, arousal, and dominance. In our case, the ANEW score stands for the rating in terms of valence. The positive and negative dimensions represented the number of positive/negative sentiment words found again, as described previously, which were also found in the Bing Liu Opinion Lexicon (Hu & Liu, 2004).

The algorithm for dealing with negation was as follows. If among the children of a head of a dependency graph was a negating word (e.g.: not), then all the words in that dependency graph were considered negated, which impacted the way the sentiment score was updated. If a sentiment score from ANEW was found, then the anew score was updated with $7 - word\_anew\_score$. If a positive word was found, then the counter for negative words was increased. If a negative word was found, then the counter for positive words was increased.

For a review, I computed the sentiment score per topic in this manner only for the topics that had a probability of appearance as outputted by the LDA model of at least 10%. Otherwise, the score was 0 by default on all dimensions, considering that the topic does not appear or has too little influence.

In the end, I represented each review as an array of dimension 100, where the first 25 cells corresponded to the topic distribution of the review and the remaining 75 represented the associated 25 sentiment scores per review, where a sentiment score is 3-dimensional.

5. Classification Models

The features presented previously were inputted to the following classification models. While the Multinomial Naïve Bayes, Passive Aggressive classifier and Support Vector Machines were chosen more as baseline methods, but also as means of observation for the effects of different types of pre-processing techniques, the neural networks were chosen in hope that the added complexity would lead to an improvement in classification accuracy. In particular, the use of convolutional networks is meant to test the hypothesis that the same local patterns principles that work when it comes to images are valuable in the case of natural language as well, where proximity now represents the context of words.

5.1. The Multinomial Naïve Bayes

The Multinomial Naïve Bayes classifier is still one of the most popular baseline methods for text categorization having word frequencies as input. It belongs to the family of probabilistic methods, trying to estimate for a document d the probability of belonging to each class c, given its features:

$$P(c|d) \stackrel{\text{def}}{=} \frac{score(d,c)}{\sum_{c' \in C} score(d,c')} \tag{2.5.5}$$

where

$$score(d,c) \stackrel{\text{def}}{=} P(c) * \prod_{i=1}^{n} P(f_i|c) \tag{2.5.6}$$

represents the score of belonging to the class c, f_i are the features, $P(c)$ is the prior probability of each class, which is determined by checking the frequency of each class in the training set, and $P(f_i|c)$ represents the contribution of a feature toward the class likelihood for a class. In the end, the document is assigned to the class having the maximum score. In this computation, however, the score becomes 0 when a feature never occurs with a given label in the training set. This problem is addressed by applying soothing techniques, given the assumption that because a feature/label combination did not occur in the training set, does not mean it is impossible for that combination to occur.

The main disadvantage of this classifier is the naïve assumption that the features are independent one from another. For example, if one feature is “best” and another one is “world’s best”, then their probabilities are going to be multiplied as if they are independent when in fact they are not (they have similar meaning) (Potts, 2011).

5.2. Support Vector Machines

A linear classifier is a classifier that assumes the classes it has to discern are linearly separable and tries to find the plan that separates them. To do so, it employs the following function:

$$f(x_i, W, b) = Wx_i + b \tag{2.5.7}$$

that maps each example x_i to a confidence score for each class. W and b are the parameters of the model. W is the vector of weights, while b represents the bias. An interpretation for the W weights is that each row of W corresponds to a learned template for one of the classes.

Multiclass Support Vector Machine (SVM) is a type of linear classifier that has a specific loss function that tries to map an example to a vector of scores in which the correct class is given a higher score than the incorrect classes by some fixed margin Δ. Formally, this loss function is defined as follows:

$$L = \frac{1}{N}\sum_i L_i + \lambda R(W) \tag{2.5.8}$$

where L_i is the loss function for the i-th example x_i that has associated the correct class y_i:

$$L_i = \sum_{j \neq y_i} max(0, f(x_i, W, b)_j - f(x_i, W, b)_{y_i} + \Delta) \tag{2.5.9}$$

and R(W) is the regularization penalty that ensures the weights found are unique and correspond to the best separating plan:

$$\mathrm{R(W)} = \sum_k \sum_l W_{k,l}^2 \tag{2.5.10}$$

(Karpathy, Linear Classification: Support Vector Machine, Softmax, 2016)

To find the weights, the *gradient descent method* is used, which implies starting from a set of weights in the parameter space and then updating them using the following rule: $w_i = w_i - \alpha * \frac{\partial}{\partial w_i} Loss(w),\ w_i \in w$, where $Loss$ is the loss function and α is the *learning rate.* The learning rate can be either constant or it can decay as the learning advances.

5.3. Passive Aggressive Classifier

The Passive Aggressive Classifier belongs to the class of online training algorithms proposed by Crammer et al. (2006), which updates its weights after each example is presented, unlike the SVM, where the examples come in batch. Its name comes from its behaviour when examining an example. If the example is correctly classified, then the algorithm is passive (the weights are not updated). Otherwise, the update rule becomes very aggressive since it must find new weights that are close to the previous ones (so that the existing knowledge is not lost) and at the same time determine a correct classification of the example (to some degree of tolerance). Being a linear classifier, the classification is done in the same approach: by mapping each example x_i to a confidence score for each class and choosing the class with the highest score.

The loss function for an example is defined as:

$$\ell(w(i)\,;\,(x_i\,,y_i)) \;=\; \max_{u\neq y_i}(0, 1 \;-\; (\,w(i)y_i \;\cdot\; x_i \;-\; w(i)u \;\cdot\; x_i)) \qquad (2.5.11)$$

and the new weights are given by solving the following optimization function:

$$arg\,\min_{w} \tfrac{1}{2}\textstyle\sum_{u=1}^{K}\left\|w_u - w_u^{(i)}\right\|^2 + C\xi^2 \;\; s.t.\,(wy_i \;\cdot\; x_i \;-\; w_{p_i} \;\cdot\; x_i) \;\geq\; 1 - \xi \qquad (2.5.12)$$

where ξ is a slack variable and C is a parameter that controls its degree of influence (Matsushima, Shimizu, Yoshida, Ninomiya, & Nakahawa, 2010).

5.4. Neural networks

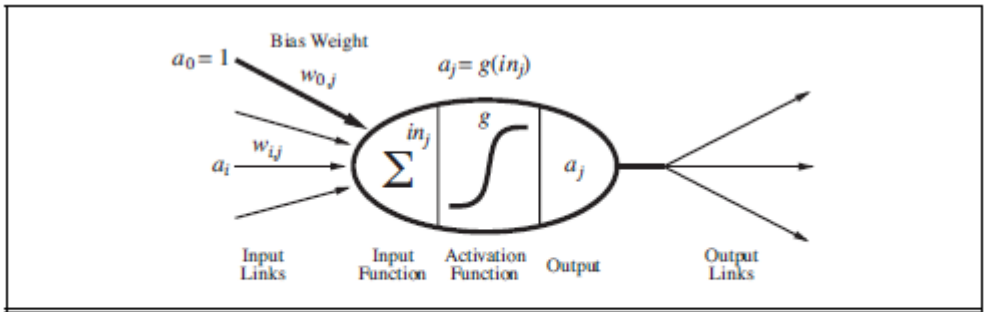

Figure 8. The perceptron (McCulloch & Pitts, 1943 cited in Russell & Norvig, 2010)

Neural networks are mathematical models implemented by computers inspired by the neuroscience theory of the human brain's activity being based on the electrochemical activity of the brain cells (neurons). In artificial intelligence, the neuron is modelled as in Figure 8.

A neural network contains many such neurons connected by direct links. The output of a neuron j is $a_j = g(\sum_{i=0}^{n} w_{i,j} * a_i)$, where g is the activation function, a_i is the input for neuron j and at the same time the output of a neuron i with whom the neuron j has a direct link that allows for direct propagation, $w_{i,j}$ are the weights associated with this link, $i = \{1, .., n\}$, $a_0 = 1$ represents bias and $w_{0,j}$ is its weight.

There are two ways of connecting the neurons in a neural network, which leads to the existence of two types of networks:

- *Recurrent neural networks,* in which the output of a neuron is used as an input by the same neuron
- *Feed-forward*, in which the connections between neurons form an oriented acyclic graph; the neurons are in general organized in layers with only neurons on adjacent layers being able to communicate (Russell & Norvig, 2010)

5.4.1. Feed-forward neural networks

As stated before, supervised learning comes down to finding a hypothesis that minimizes the loss function. In the case of neural networks, finding that function is equivalent to finding the weights associated with the links between neurons in such a way that the loss function reaches a minimum. To find those weights, the same *gradient descent method* as in SVM is used.

Feed-forward neural networks are used for learning non-linear, more complex patterns. They are formed of an input layer, an output layer and one or more hidden layers. The loss function, however, is known only for the output layer. As a result, the error from the last layer propagates back to the neurons on the layers before in order to update their weights in the same manner, based on the idea that the j^{th} neuron is responsible for a part of the error associated with each of the neurons on the following layer with whom it is connected. This algorithm is called *back-propagation.*

The *ReLU (Rectified Linear Unit)* activation function has the following form:

$$f(x) = \max(0, x) \tag{2.5.13}$$

It has become one of the most popular activation functions for the hidden layers due to its advantages:

- it strongly accelerates the convergence of the stochastic gradient descent as compared to the hyperbolic tangent or sigmoid
- it requires less expensive operations than the hyperbolic tangent or sigmoid

However, it has the disadvantage of being rather fragile during training or even "die" if, for example, the learning rate is too large.

For the neurons on the output layer the *softmax* transfer function is a popular choice in classification problems. The function has the following form:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \qquad (2.5.14)$$

It takes an array of real values and it compresses it into a vector with values between 0 and 1 having the sum 1 (Figure 9). The output can be interpreted as the (normalized) probability of class j, given the array x_i and parameterized by the weight matrix W:

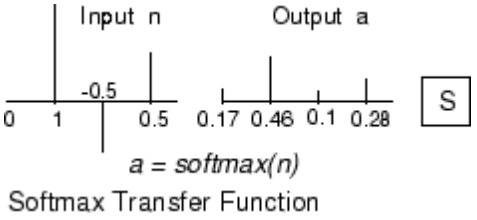

Figure 9. The softmax transfer function (Mathworks, 2006)

$$P(y_i|x_i; W) = \frac{e^{f_{y_i}}}{\sum_j f_j} \qquad (2.5.15)$$

The function interprets the scores $f(x_i; W, b) = W * x_i + b$ as unnormalized log probabilities. Exponentiating gives the (unnormalized) probabilities and the division to the sum performs the normalization.

The softmax classifier is minimizing the *cross-entropy loss function*, which strongly penalizes the results that are far from the right class:

$$L_i = -\log\frac{e^{f_{y_i}}}{\sum_j f_j} \text{ sau echivalent } L_i = -f_{y_i} + \log\sum_j e^{f_j}. \qquad (2.5.16)$$

In general, the cross-entropy between the real distribution p and the estimated distribution q is defined as follows:

$$H(p, q) = -\sum_j p(x) \log q(x) \qquad (2.5.17)$$

In our case, q is the distribution of the estimated class probabilities ($q = \frac{e^{f_{y_i}}}{\sum_j f_j}$), while p is the distribution in which the entire probability is assigned to the right class ($p = [0,0, \dots 1 \dots 0]$ has only one 1 on the position corresponding to y_i).

The cross-entropy can be written as a function of entropy and the Kullback-Leibler divergence: $H(p,q) = H(p) + D_{KL}(p||q)$. Since the entropy of the delta p is zero, minimizing the loss function is equivalent to minimizing the divergence between the two distributions (i.e. the softmax function having as output the entire probability on the position of the right class). In probabilistic interpretation, minimizing the objective function (the loss function) is equivalent with minimizing the negative log likelihood of the right class, which can be interpreted as performing a Maximum Likelihood Estimation (Karpathy, 2016).

During the experiments, I employed the following feed-forward neural networks models as a way of learning the linguistical patterns of expressing different sentiments by humans.

5.4.1.1. Fully-connected feed-forward neural network on general word embeddings

The chosen architecture for this model was:

- **an input layer** that takes the **GloVe embeddings** of the words belonging to each document (a matrix of size 1234x50, where 1234 is the maximum number of words in a document and 50 is the dimensionality of the GloVe embedding; if a document had less than 1234 words, zero-padding is applied)
- **a hidden layer** with **128 neurons** that have **ReLu** as the activation function
- **a dropout layer** that randomly sets **0.2** of input units to 0 at each update during training time
- and **an output layer with 5 neurons** with **softmax** as an activation function.

By only keeping a neuron active with some probability p (which is equal to 0.8 in this case), the dropout layer prevents overfitting.

5.4.1.2. Fully-connected feed-forward neural network on sentiment word embeddings

The architecture for this model was:

- **an input layer** that takes the **GloVe embeddings** of the words belonging to each document (a matrix of size 1234x50**). The embeddings** act as weights **and are trainable.**
- and **an output layer with 5 neurons** with **softmax** as an activation function.

5.4.1.3. Fully-connected feed-forward network on handcrafted topic features

In order to assess the singular value of the handcrafted topic features, I fed them to a feed-forward network. This model represented in fact a softmax linear classifier, that took the handcrafted features without further processing them via hidden layers and learned its weights. The architecture of the model was the following:

- **an input layer** taking in the **handcrafted topic features** (i.e. the topic distributions of the reviews – an array of 25 elements)
- an **output softmax layer** with **5 neurons**

5.4.1.4. Fully-connected feed-forward network on handcrafted sentiment topic features

The handcrafted sentiment topic features were added as well to a softmax classifier for an assessment of their informational value. The architecture of the model was similar to the one in the previous section:

- an **input layer** taking in the **handcrafted sentiment topic features** (i.e. the topic distributions of the reviews and the sentiment scores per topic)
- **a softmax layer with 5 neurons**

5.4.2. Convolutional networks

Convolutional networks are a type of feed-forward neural networks that were created as a response to the very large number of parameters that had to be learnt when dealing with high-resolution images. Thus, they were especially designed for image processing and were inspired by the animal visual system.

Unlike classic feed-forward neural networks, convolutional ones organize the neurons in a layer in three dimensions: width, height and depth in order to process data from a multi-dimensional space. Regarding architecture, the first layer is the input layer, which is generally followed by convolutional layers, ReLu layers, pooling layers and in the end fully-connected

layers. An example of a classic architecture for a convolutional network applied in natural language processing is depicted in Figure 10.

By using convolutional and pooling layers before fully-connected ones, a convolutional network reduces the number of needed parameters. Moreover, such a network takes into account the spatial relationship between data and treats points that are closely located different from the ones that are far from each other.

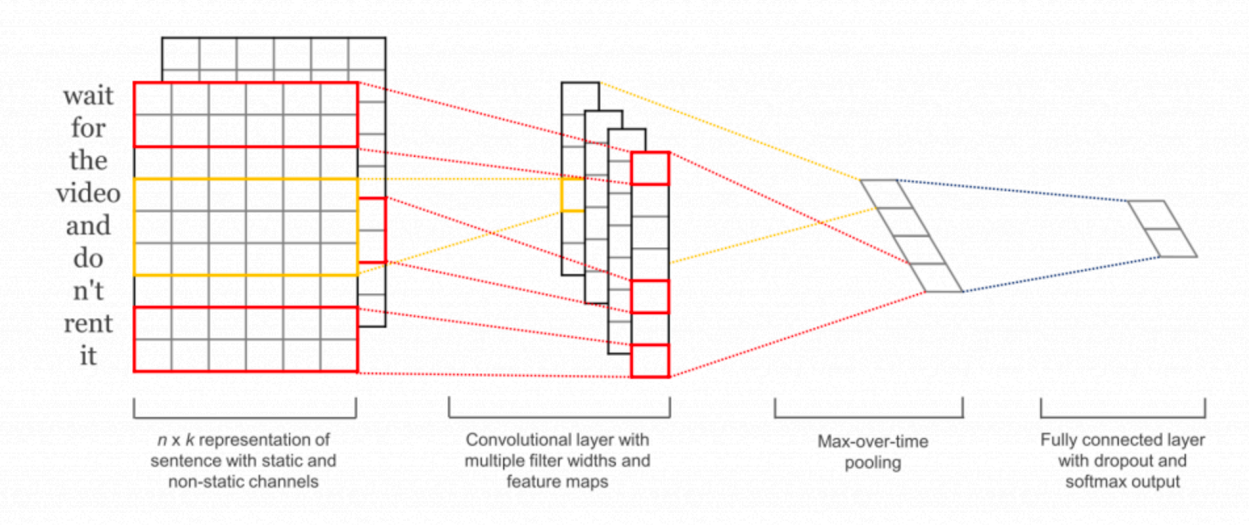

Figure 10. Convolutional network for sentence classification (Kim, 2014)

The convolutional layer, that gives the network's name, is perhaps the most important layer of the architecture. It contains K filters that can be learned. For natural language processing, where the depth dimension is missing since a document is represented as a matrix, with each row corresponding to a word in the form of a real-valued array (word embeddings) or a one-hot vector that index the said word in a vocabulary, a 1D convolution layer is needed. This means that the convolutional operation is performed on only one dimension. Thus, while the height of the filter (F) is usually reduced to a small fraction of the height of the input matrix (common values are 2 to 5), the width corresponds to its full width. Performing convolution over the input data of size $W_1 \times H_1$ means computing the dot product between each filter and each window – corresponding to the filter dimension ($F \times H_1$) and the stride (S) that gives the distance between the starting points of two windows (to which zero-padding – P can eventually be applied). The result is represented by a unidimensional activation map for each filter that represents the response of the filter to each window. The output of the layer is represented by this list of activation maps. Intuitively, the

learned filters in the first layer could be said to capture features quite similar (but not limited) to n-grams.

The purpose of the pooling layer is to reduce the number of parameters (and thus, computations needed) in the network and prevent overfitting. The most common way to do it is to apply a max operation to the result of each filter and thus keep the most important feature in the feature map (Karpathy, 2016).

Given that the discovery of local patterns performed by convolutional networks led to such good results in computer vision problems and starting from the assumptions that proximity is also important in the natural language discourse, different architectures of convolutional networks responding to different inputs were employed in different experiments:

5.4.2.1. Convolutional network on general word embeddings

The convolutional network I employed follows the same classical recipe depicted in Figure 10, having:

- **an input layer** that takes the **GloVe embeddings** of the words belonging to each document (a matrix of size 1234x50)
- **one convolutional layer** with 128 filters, stride of 1, zero-padding and ReLu as the activation function
- **a global max-pooling layer**
- and the final layer - **a fully connected softmax layer** whose output is the probability distribution over labels

The filter size for the convolutional layer was determined experimentally, by continuously incrementing it until the accuracy on the validation set no longer improved. As it turns out, 7 is the number that best leverages closeness and context. Lower values represent window sizes that capture too general expressions, while higher ones capture too much particularity and quickly lead

to overfitting. The training evolution for the models with different filter size are depicted in Figures 11-14.

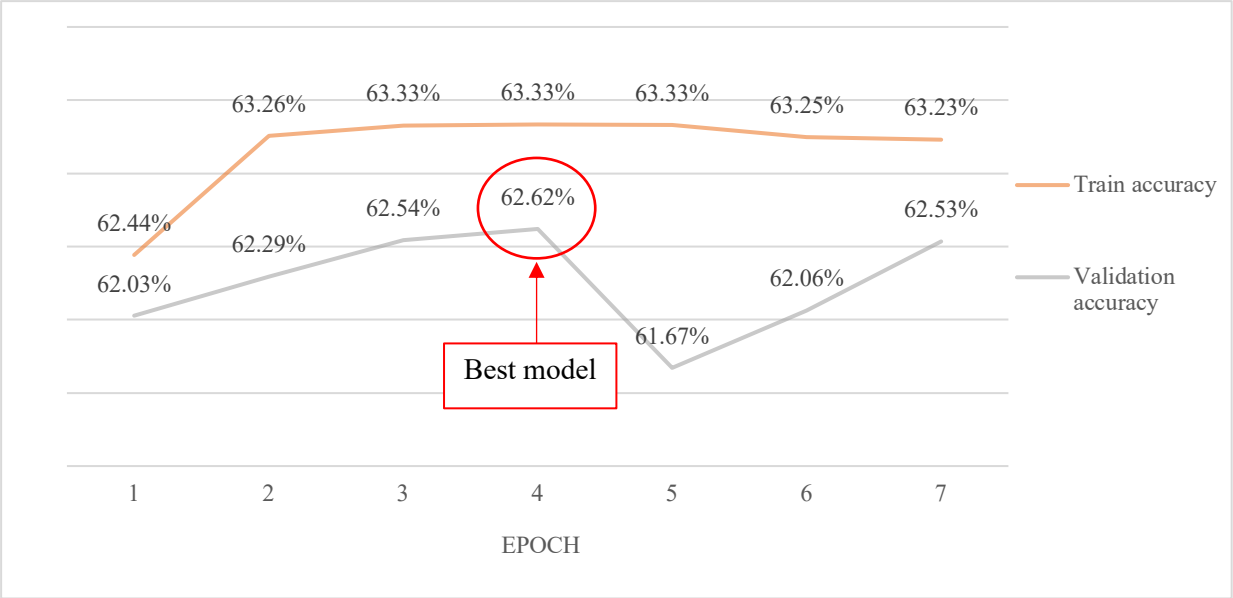

Figure 11. Accuracy evolution for convolutional network with filter_size=3

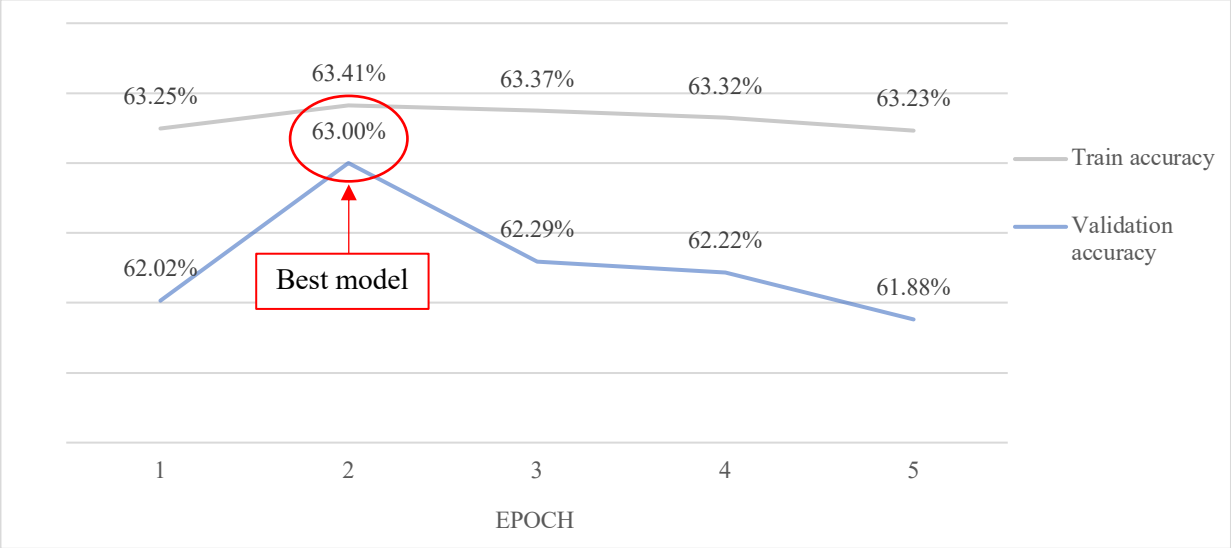

Figure 12. Accuracy evolution for convolutional network with filter_size=5

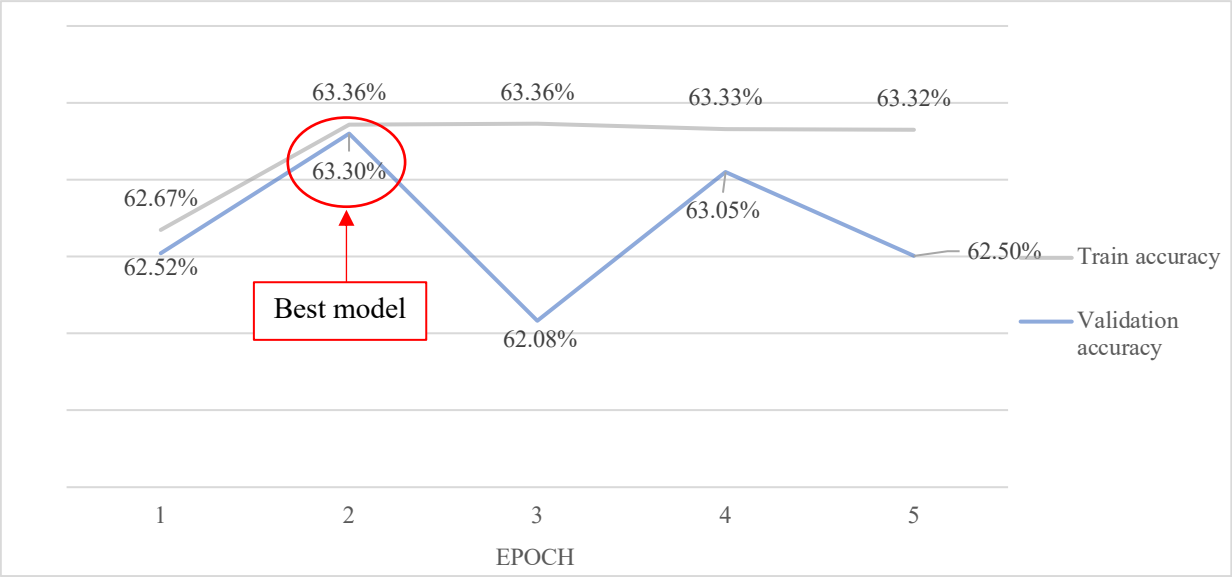

Figure 13. Accuracy evolution for convolutional network with filter_size=7

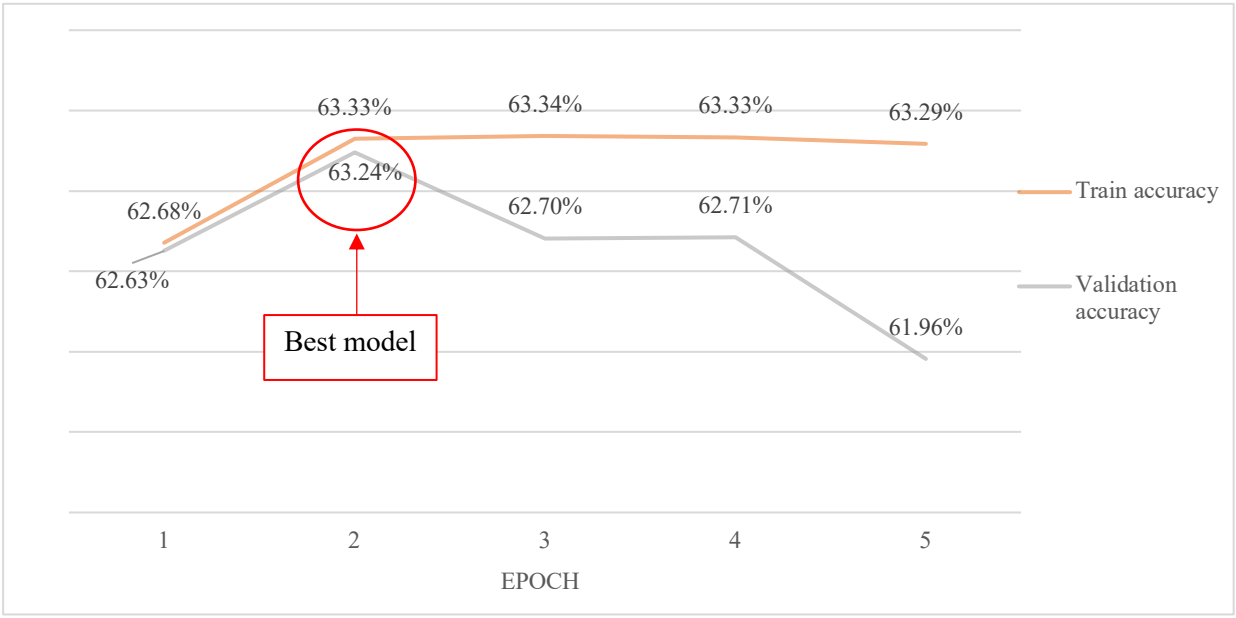

Figure 14. Accuracy evolution for convolutional network with filter_size=9

5.4.2.2. Convolutional network on restaurant sentiment embeddings

The restaurant sentiment embeddings are the result of allowing the convolutional neural network presented in the previous section to adjust the word embeddings in response to the classification feedback. The automatic features created in the learning process via the combination

of convolutional and max-pooling layers are finally fed to the same softmax layer that assigns a review to a sentiment.

Due to the fact that learning the embeddings may lead to overfitting, a dropout layer was added to the model with filter size 7 before the final softmax layer. The dropout layer is supposed to reduce overfitting by randomly deactivating neurons. However, in this case, it only reduced accuracy on test set (Figure 15). As it turns out, the activation maps outputted by the convolutional and max pooling layers of the convolutional network are all important for classification.

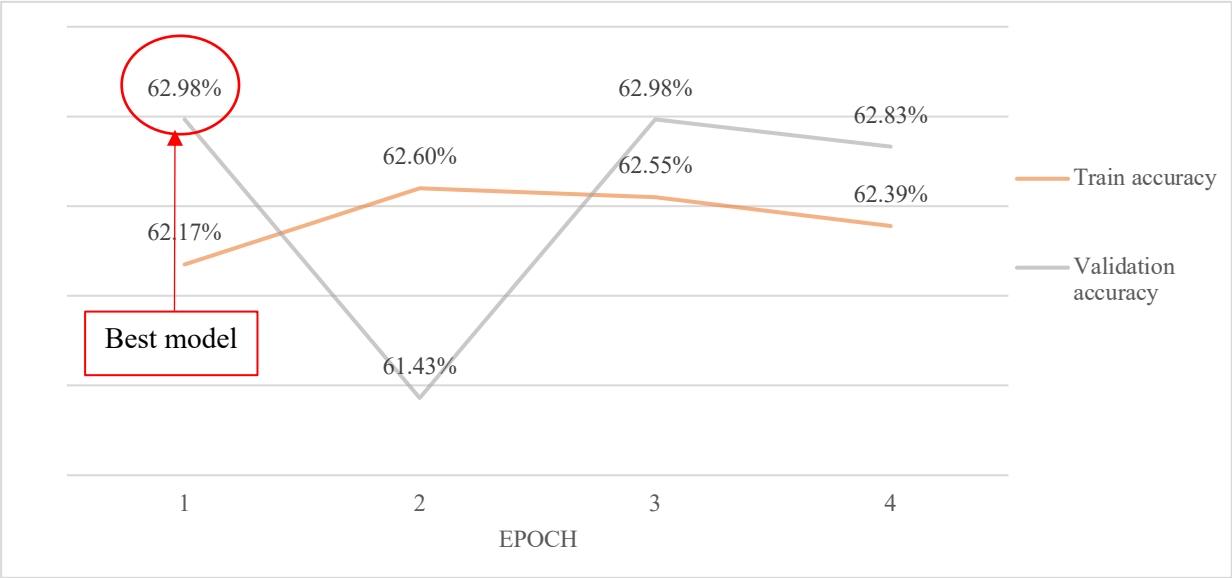

Figure 15. Accuracy evolution for convolutional network with dropout layer

5.4.2.3. Neural network on sentiment embeddings and handcrafted topic features

Assuming that people feel differently about different aspects of the restaurant experience and that, by adding the topic distribution into the review representation, the model could be provided with the necessary information to learn the association between the overall sentiment of a review and the topics it contains, a new model with two sources of input: the sentiment embeddings and handcrafted topic features was built. The sentiment embeddings were fed to the best convolutional network so far (i.e. the one with filter size 7) that was stripped off its final softmax layer and its output was concatenated with the topic distribution

so that it could be further passed on to the final softmax layer making the classification. The model architecture is presented below:

- **an input layer** that takes the **GloVe embeddings** of the words belonging to each document (a matrix of size 1234x50)
- **one convolutional layer** with 128 filters, stride of 1, zero-padding and ReLu as the activation function
- **a global max-pooling layer**
- a layer that **concatenates the output of the previous ones with the handcrafted topic features**
- a **fully-connected softmax layer** with 5 neurons

5.4.2.4. Neural network on sentiment embeddings and handcrafted sentiment topic features

To further add information about the sentiment regarding each topic, I computed sentiment topic features as described previously and added it to a model similar to the one in the previous section, only now with three sources of input: sentiment embeddings, topic distribution and sentiment topic features. Again, sentiment embeddings are fed to the convolutional network with filter size 7 from section stripped off its final classification layer and its output is concatenated with topic distribution and sentiment topic features to be fed to a final softmax layer. A summary of the model is presented below:

- **an input layer** that takes the **GloVe embeddings** of the words belonging to each document (a matrix of size 1234x50, where 1234 is the maximum number of words in a document and 50 is the dimensionality of the GloVe embedding)
- **one convolutional layer** with 128 filters, stride of 1, zero-padding and ReLu as the activation function
- **a global max-pooling layer**
- a layer that **concatenates the output of the previous ones with the handcrafted sentiment topic features**
- a **fully-connected softmax layer** with 5 neurons

The training procedure was the same for all neural networks presented above (both convolutional and regular feed-forward). The dataset was divided into train, validation and test set according to this ratio: 64-16-20. This resulted in a train set of 2,061,708 examples, a validation set of 515,428 examples and a test set of 644,283 examples. At the end of each training epoch, the trained model was evaluated on the validation test. If the accuracy did not improve for three epochs in a row, then the training stopped and the model corresponding to the best performance so far was saved. This technique is called early-stopping. The loss function was the categorical cross-entropy and the metric to compare models on was the accuracy. In order to optimize the gradient descent algorithm, the RMSprop method was used. RMSprop is an adaptive learning rate method that divides the learning rate by an exponentially decaying average of squared gradients.

6. Classification Results

6.1. Classification performed on word counts

The result of the classification using the **Naïve Bayes Classifier, Support Vector Machines** and the **Passive Aggressive Classifier** to which **word counts** were fed are illustrated in Table 7.

As it turns out, removing or not stop words or punctuation and performing or not stemming on the reviews does not have any impact whatsoever on the performance of the Naïve Bayes classifier, which stays around 0.59.

The overall performance of the Support Vector Machines is weaker than the one of the first classifier (F1-score is in general lower). Here, however the results seem to be mildly positively influenced by removing the stopwords, the punctuation and performing stemming, since the performance is weaker when none of these transformations are performed.

Initially, the dataset was not balanced in order to preserve the real distribution of the scores. However, in this setup, there was a significant difference between the recall values for the 2 and 5 stars classes which meant that a lot more 5 stars reviews were recognized as such compared to 2 stars reviews. Thus, the training of the SVM Classifier was performed again on a set that was balanced by random under sampling. The testing, however, was still done on data preserving the real distribution. The results can be observed in the following table. Overall, the precision improved, the recall values were more evenly balanced between classes, and the F1-score was higher. In the case of both balanced and unbalanced data, there is an obvious drop in performance when removing the stopwords from the dataset, and also a slight increase when performing stemming, peaking even above the Naïve Bayes score.

The **Passive Aggressive Classifier** has the lowest performance of all classifiers on word counts. Here, the best result of the F1-score (0.55) is obtained using the features that contained stopwords, punctuation and no stemming was performed.

Table 7. Classification performed on word counts - Results

Classifier		Naïve Bayes Classifier			Support Vector Machines			Support Vector Machines – trained on balanced dataset			Passive Aggressive Classifier		
Features / Performance		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Word counts – set 1 – stop words and punctuation removal													
Class	1	0.6	0.72	0.65	0.64	0.84	0.72	0.61	0.82	0.7	0.67	0.59	0.63
	2	0.4	0.34	0.36	0.45	0.22	0.29	0.4	0.41	0.41	0.33	0.29	0.31
	3	0.45	0.39	0.42	0.5	0.28	0.36	0.44	0.41	0.42	0.31	0.43	0.36
	4	0.53	0.49	0.51	0.53	0.39	0.45	0.58	0.35	0.43	0.44	0.4	0.42
	5	0.72	0.79	0.75	0.65	0.9	0.75	0.7	0.84	0.77	0.69	0.68	0.68
Average		0.59	0.6	0.59	0.57	0.6	0.57	0.59	0.6	0.58	0.53	0.52	0.52
Word counts – set 2 – stop words and punctuation removal and stemming													
Class	1	0.59	0.71	0.64	0.63	0.84	0.72	0.6	0.82	0.69	0.62	0.62	0.62
	2	0.39	0.33	0.36	0.44	0.22	0.29	0.34	0.46	0.39	0.33	0.28	0.3
	3	0.44	0.38	0.41	0.5	0.23	0.31	0.43	0.32	0.37	0.31	0.36	0.34
	4	0.53	0.48	0.5	0.51	0.43	0.47	0.57	0.35	0.44	0.44	0.37	0.4
	5	0.71	0.79	0.75	0.66	0.88	0.75	0.71	0.83	0.76	0.65	0.72	0.69
Average		0.58	0.59	0.58	0.57	0.6	0.57	0.58	0.59	0.57	0.51	0.52	0.52
Word counts – set 3 – punctuation removal													

Class	1	0.6	0.7	0.65	0.66	0.82	0.73	0.68	0.73	0.7	0.55	0.78	0.64
	2	0.39	0.34	0.36	0.42	0.4	0.41	0.35	0.64	0.45	0.37	0.26	0.31
	3	0.45	0.41	0.42	0.56	0.24	0.34	0.47	0.25	0.33	0.47	0.16	0.24
	4	0.53	0.49	0.51	0.56	0.37	0.44	0.57	0.44	0.49	0.46	0.48	0.47
	5	0.73	0.78	0.75	0.65	0.91	0.76	0.73	0.81	0.76	0.66	0.78	0.72
Average		0.59	0.6	0.59	0.59	0.61	0.58	**0.61**	**0.60**	**0.59**	0.54	0.56	0.53
Word counts – set 4 – stopwords removal													
Class	1	0.6	0.7	0.65	0.71	0.74	0.73	0.63	0.82	0.72	0.78	0.47	0.58
	2	0.38	0.34	0.36	0.46	0.39	0.42	0.45	0.3	0.36	0.33	0.45	0.38
	3	0.45	0.41	0.43	0.54	0.29	0.38	0.4	0.59	0.48	0.37	0.33	0.35
	4	0.54	0.49	0.51	0.55	0.34	0.42	0.63	0.17	0.26	0.48	0.37	0.42
	5	0.72	0.78	0.75	0.63	0.93	0.75	0.66	0.9	0.76	0.66	0.81	0.73
Average		0.59	0.6	0.59	0.59	0.61	0.57	0.59	0.59	0.54	0.55	0.55	0.54
Word counts – set 5													
Class	1	0.6	0.71	0.65	0.7	0.77	0.74	0.78	0.55	0.65	0.67	0.7	0.69
	2	0.4	0.32	0.36	0.55	0.17	0.26	0.37	0.62	0.47	0.42	0.17	0.24
	3	0.45	0.42	0.43	0.56	0.15	0.23	0.38	0.51	0.43	0.4	0.45	0.42
	4	0.54	0.49	0.51	0.44	0.64	0.52	0.56	0.38	0.45	0.45	0.51	0.48
	5	0.72	0.78	0.75	0.72	0.79	0.76	0.75	0.76	0.76	0.7	0.7	0.7
Average		**0.59**	**0.6**	**0.59**	0.6	0.6	0.56	0.61	0.59	0.59	**0.56**	**0.56**	**0.55**
Word counts – set 6 - stemming													
Class	1	0.59	0.7	0.64	0.68	0.79	0.73	0.76	0.58	0.66	0.65	0.61	0.63

	2	0.39	0.32	0.35	0.5	0.25	0.33	0.34	0.67	0.45	0.3	0.37	0.33
	3	0.45	0.41	0.43	0.5	0.33	0.4	0.42	0.43	0.43	0.31	0.55	0.39
	4	0.53	0.49	0.51	0.52	0.53	0.52	0.54	0.45	0.49	0.48	0.35	0.41
	5	0.72	0.78	0.75	0.7	0.83	0.76	0.77	0.72	0.74	0.75	0.64	0.69
Average		0.59	0.6	0.59	**0.6**	**0.62**	**0.6**	**0.62**	**0.59**	**0.59**	0.56	0.52	0.53

6.2. Classification performed on TF-IDFS

When using TF-IDFS features, the Naïve Bayes Classifier becomes the weakest classifier, having as best performance on the test set an F1-score of 0.48, on the features that had the stopwords and punctuation removed and on which no stemming was performed. Moreover, the classifier offered some extreme performances for the recall of reviews of 2 and 5 stars. While almost none of the reviews of 2 stars were actually predicted as being a 2 stars review, almost all 5 stars reviews were labelled correctly.

The Support Vector Machines performs slightly better than the Naïve Bayes Classifier. However, here the best performance is obtained when stopwords and punctuations are kept and again no stemming is performed. This could support the hypothesis that stopwords and punctuation could be an expression of subjectivity and thus indicate the opinions of the reviewer regarding the restaurant under loop. Also, we notice the presence of the same extreme recall for both 2 stars and 5 stars reviews, being gentler on the first class and sharper on the second one.

When using TF-IDFS, the best performance is reported by the Passive Aggressive classifier, on the features that included stopwords and punctuation and no stemming was performed. Also, worth noting is the fact that this online classifier is not affected by classes being unbalanced. On the contrary, it seems to thrive on it. For both word counts and TF-IDFS the results were better when preserving the actual distribution of reviews.

Table 8. Classification performed on TF-IDFs - Results

Classifier		**Naïve Bayes Classifier**			**Support Vector Machines**			**Passive Aggressive Classifier**		
Features / Performance		**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
TF-IDFS – set 1 – stop words and punctuation removal										
Class	1	0.67	0.66	0.66	0.56	0.87	0.68	0.65	0.72	0.68
	2	0.45	0.03	0.06	0.46	0.13	0.2	0.37	0.35	0.36
	3	0.32	0.06	0.1	0.52	0.17	0.25	0.4	0.34	0.37
	4	0.38	0.43	0.4	0.47	0.26	0.33	0.49	0.46	0.47
	5	0.61	0.89	0.73	0.59	0.95	0.73	0.7	0.76	0.73
Average		**0.5**	**0.54**	**0.48**	0.53	0.56	0.5	0.56	0.58	0.57
TF-IDFS – set 2 – stop words and punctuation removal and stemming										
Class	1	0.67	0.65	0.66	0.56	0.87	0.68	0.66	0.71	0.68
	2	0.44	0.03	0.06	0.47	0.12	0.19	0.38	0.31	0.34
	3	0.33	0.06	0.1	0.5	0.18	0.26	0.4	0.36	0.38
	4	0.38	0.42	0.4	0.46	0.26	0.33	0.49	0.4	0.44
	5	0.61	0.89	0.72	0.59	0.95	0.73	0.66	0.81	0.73
Average		0.5	0.54	0.47	0.53	0.56	0.5	0.55	0.57	0.56
TF-IDFS – set 3 – punctuation removal										
Class	1	0.67	0.65	0.66	0.56	0.88	0.69	0.67	0.71	0.69
	2	0.44	0.02	0.04	0.46	0.13	0.2	0.38	0.42	0.4

	3	0.29	0.04	0.08	0.53	0.19	0.28	0.46	0.31	0.37
	4	0.37	0.42	0.39	0.48	0.26	0.34	0.53	0.37	0.43
	5	0.61	0.89	0.73	0.6	0.95	0.74	0.65	0.85	0.74
Average		0.49	0.53	0.47	0.54	0.57	0.51	0.57	0.58	0.57
TF-IDFS – set 4 – stopwords removal										
Class	1	0.73	0.54	0.62	0.56	0.87	0.68	0.71	0.64	0.68
	2	0.4	0.01	0.01	0.45	0.14	0.22	0.38	0.39	0.38
	3	0.3	0.02	0.03	0.54	0.21	0.3	0.42	0.35	0.38
	4	0.33	0.37	0.35	0.49	0.25	0.33	0.49	0.49	0.49
	5	0.58	0.92	0.71	0.6	0.95	0.73	0.7	0.77	0.73
Average		0.47	0.51	0.44	0.54	0.57	0.51	0.57	0.58	0.57
TF-IDFS – set 5										
Class	1	0.74	0.5	0.6	0.56	0.87	0.69	0.72	0.66	0.69
	2	0.36	0	0.01	0.47	0.16	0.24	0.4	0.39	0.39
	3	0.3	0.01	0.03	0.56	0.16	0.25	0.4	0.46	0.43
	4	0.32	0.36	0.34	0.47	0.28	0.35	0.52	0.4	0.45
	5	0.57	0.92	0.71	0.61	0.95	0.74	0.69	0.8	0.74
Average		0.46	0.51	0.43	**0.55**	**0.57**	**0.51**	**0.58**	**0.59**	**0.58**
TF-IDFS – set 6 - stemming										
Class	1	0.75	0.48	0.68	0.56	0.87	0.68	0.65	0.77	0.7
	2	0.36	0	0.1	0.45	0.16	0.23	0.44	0.28	0.34
	3	0.3	0.01	0.03	0.55	0.19	0.28	0.48	0.23	0.31

	4	0.32	0.35	0.33	0.47	0.27	0.34	0.48	0.39	0.43
	5	0.56	0.93	0.7	0.61	0.95	0.74	0.63	0.86	0.73
Average		0.46	0.5	0.42	0.54	0.57	0.51	0.55	0.58	0.55

The existence of the extreme recall values led to the retraining of the Naïve Bayes and SVM classifiers, this time on a balanced dataset. The performance increased greatly overall. As it turns out, classification based on TF-IDFs can register lower accuracy when there is class imbalance because when there are more examples of one class, the most important word features of the frequent class risk having lower IDF, thus their best features will have a lower weight. Also, worth noticing is that while stemming does not influence the performance, punctuation or stopwords removal seem to increase the F1-score for the Naïve Bayes. The SVM classifier appears to be weaker than the Naïve Bayes trained on the same balanced dataset. Again, removing stopwords and punctuation and performing stemming do not seem to positively influence the results.

Classifier		**Naïve Bayes Classifier – trained on balanced dataset**			**Support Vector Machines – trained on a balanced dataset**		
Features / Performance		**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
TF-IDFS – set 1 – stop words and punctuation removal							
Class	1	0.59	0.68	0.63	0.52	0.91	0.66
	2	0.35	0.46	0.4	0.36	0.3	0.33
	3	0.39	0.48	0.43	0.46	0.33	0.38
	4	0.49	0.5	0.5	0.58	0.23	0.33
	5	0.79	0.61	0.69	0.65	0.89	0.75
Average		0.59	0.56	0.56	0.56	0.58	0.54
TF-IDFS – set 2 – stop words and punctuation removal and stemming							
Class	1	0.59	0.69	0.63	0.52	0.9	0.66
	2	0.35	0.46	0.4	0.37	0.27	0.31
	3	0.38	0.47	0.42	0.46	0.34	0.39
	4	0.49	0.5	0.49	0.57	0.26	0.35
	5	0.78	0.61	0.69	0.65	0.89	0.75
Average		0.58	0.55	0.56	0.56	0.58	0.54
TF-IDFS – set 3 – punctuation removal							
Class	1	0.59	0.69	0.64	0.53	0.91	0.67

	2	0.35	0.47	0.4	0.43	0.22	0.29
	3	0.38	0.49	0.43	0.44	0.5	0.47
	4	0.5	0.5	0.5	0.61	0.19	0.29
	5	0.8	0.6	0.68	0.66	0.89	0.76
Average		**0.59**	**0.55**	**0.57**	**0.58**	**0.58**	**0.54**
TF-IDFS – set 4 – stopwords removal							
Class	1	0.6	0.68	0.64	0.52	0.91	0.66
	2	0.35	0.5	0.41	0.37	0.3	0.33
	3	0.38	0.5	0.43	0.48	0.37	0.42
	4	0.49	0.51	0.5	0.6	0.23	0.33
	5	0.81	0.57	0.67	0.66	0.89	0.76
Average		**0.59**	**0.55**	**0.57**	0.57	0.58	0.54
TF-IDFS – set 5							
Class	1	0.6	0.67	0.63	0.52	0.91	0.67
	2	0.34	0.5	0.41	0.43	0.25	0.31
	3	0.38	0.51	0.43	0.46	0.49	0.47
	4	0.48	0.5	0.49	0.61	0.19	0.29
	5	0.82	0.56	0.66	0.66	0.9	0.76
Average		0.59	0.54	0.55	**0.58**	**0.58**	**0.54**
TF-IDFS – set 6 - stemming							
Class	1	0.59	0.67	0.63	0.53	0.91	0.67
	2	0.34	0.5	0.4	0.39	0.27	0.32
	3	0.37	0.51	0.43	0.44	0.48	0.46
	4	0.48	0.49	0.49	0.61	0.18	0.27
	5	0.82	0.56	0.66	0.66	0.9	0.76
Average		0.59	0.54	0.55	0.58	0.58	0.53

Contrary to the expectations, the performance of all classifiers when fed with TF-IDFS is weaker than when fed with word counts, which may imply that perhaps the words that appear in many documents should not be considered less informative for the task of sentiment analysis and that maybe the length of the review is an indicator of the position of the restaurant in the mind of

the reviewer. This hypothesis is supported by the fact that indeed the few stars reviews have on average more words than the many stars reviews.

Table 9. Average number of words per class

Class	1 star	2 stars	3 stars	4 stars	5 stars
Average number of words	144	157	150	130	101

Thus, the extra computational effort of transforming word counts into TF-IDFS does not seem to be worthwhile.

6.3. Classification performed on general word embeddings

As previously stated, the general GloVe pretrained word embeddings were fed to neural networks: more precisely to the fully connected feed-forward and convolutional networks presented in Chapter 5. Although the model is supposed to learn non-linear patterns, as it should be the case of this problem where the classes are not linearly separable, the trained fully connected feed-forward network had a similar accuracy on the test set **(60.4%)** than the best performing simpler classifiers – the Naïve Bayes and the Support Vector Machines, thus not justifying the extra complexity. On the other hand, the convolutional network offered a better accuracy on the test data - **63.59%**, implying that the automatic learning of filters and grouping the features in windows to account for context, does have a positive impact on performance. Much of the information offered by word embeddings comes from the relationships between words (expressed through distance). At the same time, not all relationships in a review hold the same importance. Usually, words used close one to another are more likely to be related in the sense of the speech. A convolutional network considers words close one to another in the document more important that the ones that are far apart. A regular feed-forward network treats all words the same. This is why the convolutional one reports better performance.

The evolution of the train and validation accuracy for the feed-forward and convolutional networks can be observed in Figures 16 and 17, respectively, while the final test results are summarized in Table 8.

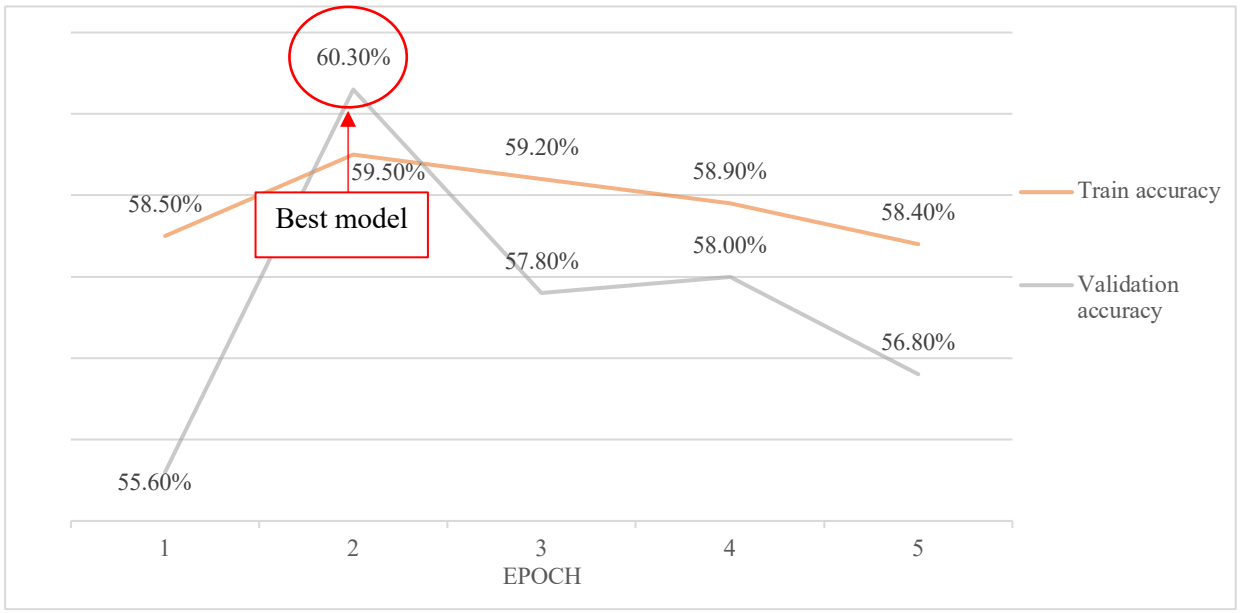

Figure 16. Accuracy evolution for feed-forward network fed with general word embeddings

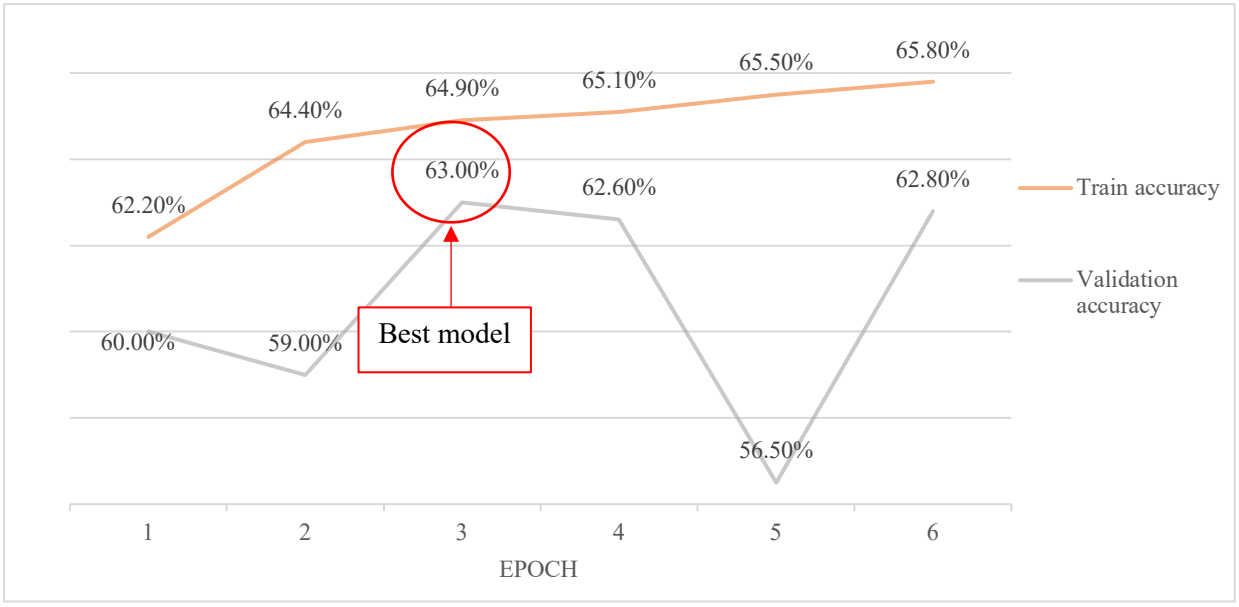

Figure 17. Accuracy evolution for convolutional network fed with general word embeddings

Table 10. Classification performed on general word embeddings - Results

Model	Test accuracy
Fully connected feed-forward neural network	60.4%
Convolutional network	63.59%

6.4. Classification performed on restaurant sentiment embeddings

Allowing the networks to learn not only the classification parameters and feature extractors, but also its embedding weights, starting from the general pretrained Glove pretrained representations and updating them in response to the classification feedback so as to better reflect restaurant review semantic has led to better classification results on the test set from both the fully

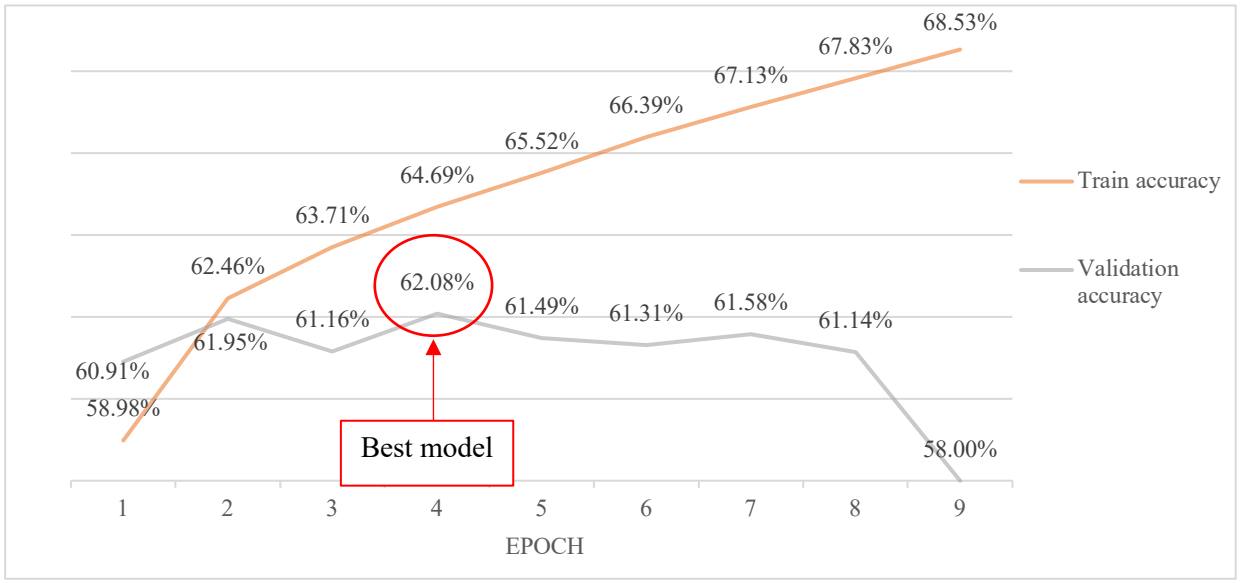

Figure 19. Accuracy evolution for feed-forward network fed with restaurant sentiment word embeddings

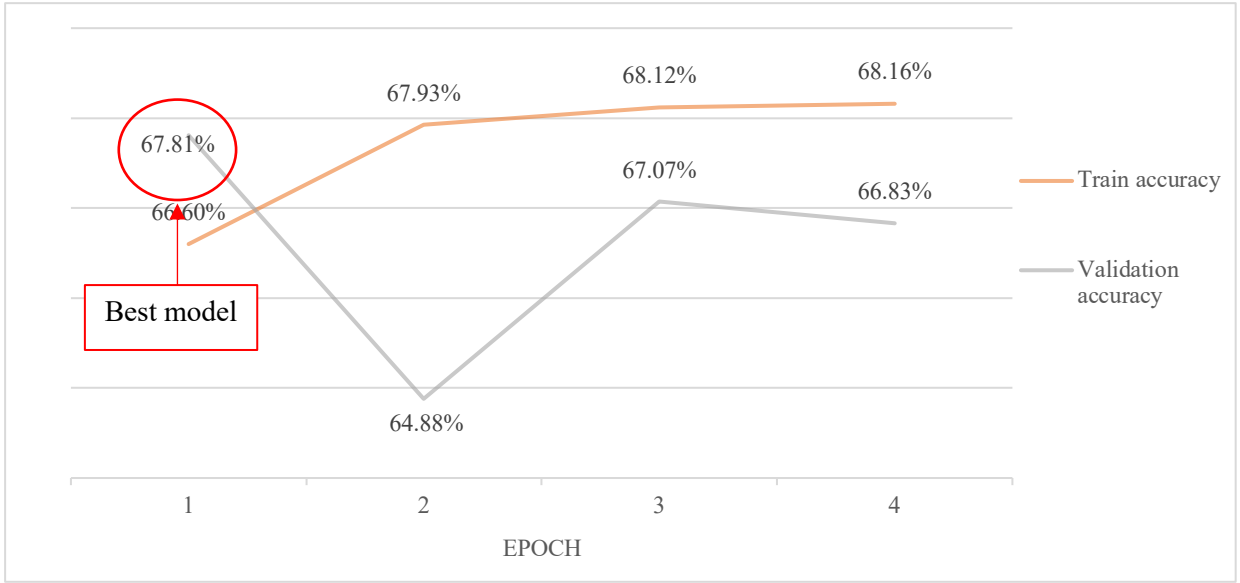

Figure 18. Accuracy evolution for convolutional network fed with restaurant sentiment word embeddings

connected feed-forward network and the convolutional one (Table 11). The training evolution of the networks can be observed in Figures 18 and 19.

Table 11. Classification performed on restaurant sentiment embeddings - Results

Model	Test accuracy
Fully connected feed-forward neural network	62.04%
Convolutional network	67.82%

6.5. Classification performed on handcrafted features

The handcrafted topic features were fed to the feed-forward neural network for informational value assessment. After training, the model registered a modest performance – only **37.78%** accuracy on the test dataset. The value being higher that 20% (the accuracy of a random classifier) proves the fact that there is some information behind these features that could help an automatic model distinguish between reviews of different opinions. However, they cannot be used independently for a state-of-the-art performance, but only complementary to other features.

Transforming the topic features into sentiment topic features did not help improve this performance. In fact, it lowered it to **37.01%** test accuracy, suggesting that the proposed handcrafted sentiment features are only noise in a machine learning context.

Table 12. Classification performed on handcrafted features - Results

Model	Test accuracy
Fully connected feed-forward neural network on handcrafted topic features	37.78%
Fully connected feed-forward neural network on handcrafted sentiment topic features	37.01%

6.6. Classification performed on sentiment embeddings and handcrafted features

The complementarity assumption proved to be wrong, at least when it came to inputting together restaurant sentiment embeddings and handcrafted topic features. The composed neural model that took in both features did not report a better accuracy than the convolutional network

fed only with restaurant sentiment embedding, but on the contrary. It achieved a test accuracy of only **65.75%** on a test set.

Neither this time the handcrafted sentiment topic features brought any improvement to the accuracy. In fact, it further decreased it to **65.01%** on the test set, supporting the hypothesis that the proposed sentiment topic features only add noise to the models.

Table 13. Classification performed on sentiment embeddings and handcrafted features - Results

Model	Test accuracy
Neural network on sentiment embeddings and handcrafted topic features	65.75%
Neural network on sentiment embeddings and handcrafted sentiment topic features	65.01%

7. Conclusions

Table 14. Results - Overall

Model	Test accuracy
Naïve Bayes on word counts	59%
Support Vector Machines on word counts	60%
Passive Aggressive Classifier on word counts	55%
Naïve Bayes on TF-IDFS	57%
Support Vector Machines on TF-IDFS	54%
Passive Aggressive Classifier on TF-IDFS	58%
Fully connected feed-forward network on general word embeddings	60.4%
Convolutional network on general word embedding	63.59%
Fully connected feed-forward network on sentiment word embeddings	62.04%
Convolutional network on sentiment embeddings	**67.82%**
Convolutional network on sentiment embeddings and handcrafted topic features	65.75%
Convolutional network on sentiment embeddings and handcrafted sentiment topic features	65.01%
Fully connected feed-forward network on handcrafted topic features	37.78%
Fully connected feed-forward network on handcrafted sentiment topic features	37.01%

Current state-of-the-art	
ULMFiT (Howard & Ruder, 2018)	70.02%
DPCNN (Johnson & Zhang, Deep Pyramid Convolutional Neural Networks for Text Categorization, 2017)	69.42%
ShallowCNN + two-views embeddings (Johnson & Zhang, Convolutional Neural Networks for Text Categorization: Shallow Word-level vs. Deep Character-level, 2016)	67.61%
Char-level CNN (Zhang, Zhao, & LeCun, 2015)	62.05%

In the end, I kept the best results per model and feature type and compared them for an overall perspective (Table 12). The best classification results were obtained using a convolutional neural network trained on restaurant sentiment embeddings, which reached 67.82% in accuracy over test data. This shows that indeed deep learning can lead us close to solving the problem of sentiment analysis, which was further complicated in this approach by trying to refine the classification into 5 classes (from 1 to 5 stars) instead of only two: positive and negative opinions. This network owes its success to not only taking into account the meaning of words by using vector spaces and further adjust it while learning, but also to preserving the local context information by processing words in windows (close to n-grams).

Moreover, as it turns out from the experiments, with a little tweaking, even simpler classifiers can have close to state-of-the-art performance, up to 60% in accuracy, on simple word counts. In fact, further processing to transform them to TF-IDFS would only lead to a decrease in performance, which suggests that in the field of sentiment analysis, the most frequent words are not necessarily less informative. This is also true for stopwords. Removing words precompiled list of frequent words in the English language (stopwords) did not lead to an increase in performance for any classifier. In fact, the Naïve Bayes classifier performed the same whether the text was additionally pre-processed or not (this includes stopwords and punctuation removal and stemming), leading to the conclusion that this pre-processing was not worthwhile. Also, another reason for which TF-IDFs may not lead to a successful classification is that the length of the review matters and is an indicator of sentiment. More positive reviews are in general shorter than negative ones. Also, the Support Vector Machines's accuracy decreased when removing stopwords.

Another thing worth noting is the fact this classifier was affected by classes being unbalanced, balancing them leading to better precision and less extreme values for recall for different classes.

All in all, there is no denying that continuous representations of words can bring more information to a model than simpler word counts, due to their incorporated semantic information. However, if they are not employed in a proper model, the gains are not that high. My experiments showed that word embeddings are not enough for boosting a model up. For example, feeding general word embeddings to a fully connected feed-forward neural network led to results similar to Naïve Bayes or Support Vector Machines trained on word counts.

However, all previously mentioned models lose context information by losing word order. A convolutional network, on the other hand, specializes in learning local patterns. Words are treated in windows and order is brought back to the table. Moreover, unnecessary noise is filtered out as only the windows with the strongest activation in relation with the classification task are kept. It is common knowledge that the meaning of a word depends on the context it is used. Context can be captured at the local level: the phrases in which a word is used and the global one, the field in which it is used, given by co-appearances of words. Thus, a sentiment analysis model can benefit from the fact that convolutional networks learns local patterns. This is obvious by the high increase in performance that appears when employing such a model. Accuracy raises again by the same amount when information about the global context is incorporated in the features through the so-called restaurant sentiment embeddings, reaching a state-of-the-art value on the fine-grained

sentiment analysis classification task. These restaurant sentiment embeddings are learned by taking general embeddings, exposing them to a restaurant review context and update them so as to optimize the same classification task.

Not the same boost in performance is brought by the use of additional handcrafted topic features. The hypothesis behind designing these features was the fact that different sentiments are associated with different aspects of the restaurant experience and these aspects are in turn associated with the different possible topics in a review. However, while the topic distribution of a review does seem to bring in some information for an automatic model, it is not enough for a satisfying discrimination between classes, as shown by the feed-forward network trained on these features. Moreover, the hypothesis that they can be used as additional features that could fine-tune a model trained on more extensive information such as the convolutional network employed was

proven wrong, as the information contained by the handcrafted features was probably contained in the filters learned by the convolutional layers, for it brought no improvement on the model.

This thesis proposes a novel approach of extracting the sentiment per topic from a review and feed it to an automatic model to learn from. It uses the topics learned by the an LDA model (the same that outputted the topic distribution of a review), and the sentiment words from the dependency graph to which the most important words for each topic belong, where sentiment words are defined by general lexicons. However, the way in which these sentiment scores are computed does not bring any value to the machine learning models in the above experiments. On the contrary, it seems to bring only noise, for the features decrease model performance when attached to the list of inputs.

By comparing the classification results when features are learned automatically versus when they are handcrafted, it becomes clear that automatic features outperform by far the handcrafted ones, for it has the advantage of learning patterns that we, humans, are not aware of, and that can be traced down by the model from real, empirical data.

8. Further work

In the context of this thesis, future directions of research could be represented by extending the variety of neural networks employed to the perhaps more popular in the NLP domain: the Recurrent Neural Networks and, in particular, the Long Term Short Term Memory network. These are models that have shown promising results in the sentiment analysis task and it would be interesting to see how they behave when fed with the restaurant sentiment embeddings in comparison with the best performing convolutional network employed in my experiments. At the same time, new recipes for automatically learning features could be derived, as it became clear from the experiments that they perform better than the handcrafted ones.

There is still a long road ahead until a computer will be able to recognize emotions in written text, let alone understand them. This is because words are intrinsically symbolic. The relationships between them are complex and the semantic spaces created so far are only mere approximations of the real mind maps. Moreover, they are cultural artefacts. Their meaning changes in relation to context or culture. They are also subjective. Their meaning can change according to the individual's own interior scale. That is why context is important or even essential if we hope to reach a perfect discrimination of sentiments. In the end, I believe that if we can find more anchors to learning the particular context of words, then we can have a chance at enabling automatic natural language processing and understanding.

Bibliography

Alvarez-Lopez, T., Juncal-Martinez, J., Fernandez-Gavilanes, M., Costa-Montenegro, E., & Gonzalez-Castaño, F. (2016). GTI at SemEval-2016 Task 5: SVM and CRF for Aspect Detection and Unsupervised Aspect-Based Sentiment Analysis. *Proceedings of SemEval-206*, (pp. 306-311).

Andreevskaia, A., & Bergler, S. (2006). Mining WordNet for fuzzy sentiment: Sentiment tag extraction from WordNet glosses. *Proceedings of the European Chapter of the Association for Computational Linguistics*, (pp. 209-216).

Asiaee, T., Tepper, M., Banerjee, A., & Sapiro, G. (2012). If you are happy and you know it... tweet. *Proceeedings of the 21st ACM international conference on Information and knowledge management*, (pp. 1602-1606).

Bakliwal, A., Arora, P., Madhappan, S., Kapre, N., Singh, M., & Varma, V. (2012). Mining sentiments from tweeets. *Proceedings of the 3rd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, (pp. 11-18).

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research, 3*, 1137-1155.

Blair-Glodensohn, S., Hannan, K., McDonald, R., Neylon, T., Reis, G., & Reynar, J. (2008). Building a Sentiment Summarizer for Local Service Reviews. *Proceedings of the WWW2008 Workshop: NLP in the Information Explosion Era.*

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 993-1022.

Crammer, K., Dekel, O., Keshet, J., Shalev-Schwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 551-585.

Dave, K., Lawrence, S., & Pennock, D. (2003). Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. *Proceedings of the 12th international conference on World Wide Web*, (pp. 519-528).

Dos Santos, C., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis on short texts. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers.*

Duda, R., Hart, P., & Stork, D. (2000). *Pattern Classification. Second Edition.* New York: Wiley Interscience.

Dumais, S. T. (2004). Latent Semantic Analysis. *Annual review of informational science and technology*, 188-230.

Esuli, A., & Sebastiani, F. (2006). SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. *Proceedings of the 5th Conference on Language Resources and Evaluation*, (pp. 417-422).

Godbole, N., Srinivasaiah, M., & Skiena, S. (2007). Large-Scale Sentiment Analysis for News and Blogs. *ICWSM, 7*(21), 219-222.

Gokulakrishnan, B., Pryanthan, P., Ragavan, T., Prasath, N., & Perera, A. (2012). Opinion mining and sentiment analysis on a twitter data stream. *Advances in ICT for Emerging Regions (ICTer), 2012 International Conference*, (pp. 182-188).

Hatzivassiloglou , V., & McKeown, K. (1997). Predicting the semantic orientation of adjectives. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eights Conference of the European Chapter of the Association for Computational linguistics*, (pp. 174-181).

Hercig, T., Brychcin, T., Svodoba, L., & Konkol, M. (2016). UWB at SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *Proceedings of SemEval-2016*, (pp. 342-349).

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification . *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 328-339). Melbourne, Australia: Association for Computational Linguistics .

Hu, M., & Liu, B. (2004). Mining and Summarizing Customer Reeviews. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 168-177).

Hu, M., & Liu, B. (2004). Mining and Summarizing Customer Reviews. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004).* Seattle.

Hu, X., Tang, J., Gao, H., & Liu, H. (2013). Unsupervised sentiment analysis with emotional signals. *Proceedings of the 22nd international conference on World Wide Web*, (pp. 607-618).

Hu, X., Tang, L., & Tang, J. (2013). Exploiting social relations for sentiment analysis in microblogging. *Proceedings of the sixth ACM international conference on Web Search and data mining*, (pp. 537-546).

Johnson, R., & Zhang, T. (2016). Convolutional Neural Networks for Text Categorization: Shallow Word-level vs. Deep Character-level. *arXiv preprint arXiv:1609.00718.*

Johnson, R., & Zhang, T. (2017). Deep Pyramid Convolutional Neural Networks for Text Categorization. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 562-570).

Jurafsky, D. (2018). *Sentiment analysis.* lecture, Stanford University, From Languages to Information.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. *Proceeding of the 52nd Annual Meeting of the Association for Computational Linguistics* , (pp. 655-665). Baltimore.

Karpathy, A. (2016). Linear Classification: Support Vector Machine, Softmax. *CS231n Convolutional Networks for Visual Recognition*.

Karpathy, A. (2016). Neural Networks Part 1: Setting up the architecture. *CS231n Convolutional Networks for Visual Recognition*. Stanford University.

Kim, S., & Hovy, E. (2006). Extracting opinions, opinion holders and topic expressed in online news media text. *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, (pp. 1-8).

Kouloumpis, E., Wilson, T., & Moore, J. (2011). Twitter sentiment analysis: The good and the bad and the omg! *Proceedings of the ICWSB.* Barcelona, Spain.

Lai , S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Twenty-ninth AAAI conference on artificial intelligence.*

Lever, J., Krzywinski, M., & Altman, N. (2016). Points of significance: Model selection and overfitting. *Nature Methods, 3*, 703-704.

Liu, B. (2015). *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions.* Cambridge: Cambridge University Press.

Martinez-Camara, E., Montejo-Raez, A., Martin-Valdivia, M., & Urena-Lopez, L. (2013). Sinai: Machine learning and emotion of the crowd for sentiment analysis in microblogs. *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013).* Atlanta, Georgia, USA.

Mathworks. (2006). Softmax transfer function [image].

Matsushima, S., Shimizu, N., Yoshida, K., Ninomiya, T., & Nakahawa, H. (2010). Exact passive-aggressive algorithm for multiclass classification using support class. *Proceedings of the 2010 SIAM International Conference on Data Mining*, (pp. 303-314).

Michael, R., Andreas, B., & Alexander, H. (2015). Exploring the Spacee of Topic Coherence Measures. *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, (pp. 399-408). Shanghai.

Mikolov , T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR.*

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Interspeech, 2*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Proceedings of the 26th International Conference on Neural Information Processing Systems*, *2*, pp. 3111-3119.

Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic Regularities for Continuous Space Word Representations. *Proceedings of NAACL-HLT*, (pp. 746-751).

Mimno, D., Wallach, H., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing semantic coherence in topic models . *EMNLP '11 Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (pp. 262-272).

Nasukawa, T., & Yi, J. (2003). Sentiment analysis: capturing favorability using natural language processing. *Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP 2003)*, (pp. 70-77). Sanibel Island.

Norvig, P., & Russel, S. J. (2010). *Artificial Intelligence: A Modern Approach. Third edition.* Upper Saddle River: Prentice Hall.

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. *Proceedings of the 7th International Workshop on Semantic Evaluation.* Association for Computational Linguistics.

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval, 2*(1-2), 1-135.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (pp. 79-86).

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representations . *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, (pp. 1532-1453).

Potts, C. (2011, November 8-9). Sentiment Symposium Tutorial. San Francisco.

Rao, D., & Ravichandran, D. (2009). Semi-supervised polarity lexicon induction. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, (pp. 675-682).

Russell, J. A. (2003). Core affect and the psychological construction of emotion. *Psychological Review, 110*(1), 145-172.

Saif, H., Fernandez, M., He, Y., & Alani, H. (2014). On stopwords, filtering and data sparsity for sentiment analysis of twitter. *Proceedings of the 9th International Language Resources and Evaluation Conference (LREC'14)*, (pp. 810-817).

Saif, H., He, Y., & Alani, H. (2012). Semantic sentiment analysis of twitter. *Proceedings of the 11th international conference on The Sematic Web - Volume Part 1 (ISWC'12)*, (pp. 508-524).

Scherer, K. (1984). Emotion as a multicomponent process: A model and some cross-cultural data. *Review of Personality & Social Psychology, 5*, 37-63.

Speriosu, M., Sudan, N., Upadhyay, S., & Baldridge, J. (2011). Twitter polarity classification with label propagation over lexical links and the follower graph. *Proceedings of the EMNLP First Workshop on Unsupervised Learning in NLP.* Edinburg.

Toh, Z., & Jian, S. (2016). NLANGP at SemEval-2016 Task 5: Improving Aspect Based Sentiment Analysis using Neural Network features. *Proceedings of SemEval-2016*, (pp. 282-288).

Turney, P. (2002). Thumbs Up or Thumbs Down? Sematic Orientation Applied to Unsupervised Classification of Reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, (pp. 417-424).

Zhang, L., Jia, Y., Zhou, B., & Han, Y. (2012). Microblogging sentiment analysis using emotional vector. *Cloud and Green Computing (CGC), 2012 Second International Conference*, (pp. 430-433).

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. *Advances in neural information processing systems*, 649-657.

Zhila, A., Yih, W., Meek, C., & Mikolov, T. (2013). Combining Heterogeneous Models for Measuring Relational Similarity. *Proceedings of NAACL-HLT*, (pp. 1000-1009).

Appendix

Appendix 1

- business.json – contains entries of the following form:

```
{
    // string, 22 character unique string business id
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",

    // string, the business's name
    "name": "Garaje",

    // string, the neighborhood's name
    "neighborhood": "SoMa",

    // string, the full address of the business
    "address": "475 3rd St",

    // string, the city
    "city": "San Francisco",

    // string, 2 character state code, if applicable
    "state": "CA",

    // string, the postal code
    "postal code": "94107",

    // float, latitude
    "latitude": 37.7817529521,

    // float, longitude
    "longitude": -122.39612197,

    // float, star rating, rounded to half-stars
    "stars": 4.5,

    // interger, number of reviews
    "review_count": 1198,

    // integer, 0 or 1 for closed or open, respectively
    "is_open": 1,

    // object, business attributes to values. note: some attribute values
might be objects
    "attributes": {
```

```
        "RestaurantsTakeOut": true,
        "BusinessParking": {
            "garage": false,
            "street": true,
            "validated": false,
            "lot": false,
            "valet": false
        },
    },

    // an array of strings of business categories
    "categories": [
        "Mexican",
        "Burgers",
        "Gastropubs"
    ],

    // an object of key day to value hours, hours are using a 24hr clock
    "hours": {
        "Monday": "10:00-21:00",
        "Tuesday": "10:00-21:00",
        "Friday": "10:00-21:00",
        "Wednesday": "10:00-21:00",
        "Thursday": "10:00-21:00",
        "Sunday": "11:00-18:00",
        "Saturday": "10:00-21:00"
    }
}
```

- review.json – contains entries of the following form

```
{
    // string, 22 character unique review id
    "review_id": "zdSx_SD6obEhz9VrW9uAWA",

    // string, 22 character unique user id, maps to the user in user.json
    "user_id": "Ha3iJu77CxlrFm-vQRs_8g",

    // string, 22 character business id, maps to business in business.json
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",

    // integer, star rating
    "stars": 4,

    // string, date formatted YYYY-MM-DD
    "date": "2016-03-09",

    // string, the review itself
```

```
    "text": "Great place to hang out after work: the prices are decent, and
the ambience is fun. It's a bit loud, but very lively. The staff is friendly,
and the food is good. They have a good selection of drinks.",

    // integer, number of useful votes received
    "useful": 0,

    // integer, number of funny votes received
    "funny": 0,

    // integer, number of cool votes received
    "cool": 0
}
```

- user.json – contains entries of the following form

```
{
    // string, 22 character unique user id, maps to the user in user.json
    "user_id": "Ha3iJu77CxlrFm-vQRs_8g",

    // string, the user's first name
    "name": "Sebastien",

    // integer, the number of reviews they've written
    "review_count": 56,

    // string, when the user joined Yelp, formatted like YYYY-MM-DD
    "yelping_since": "2011-01-01",

    // array of strings, an array of the user's friend as user_ids
    "friends": [
        "wqoXYLWmpkEH0YvTmHBsJQ",
        "KUXLLiJGrjtSsapmxmpvTA",
        "6e9rJKQC3n0RSKyHLViL-Q"
    ],

    // integer, number of useful votes sent by the user
    "useful": 21,

    // integer, number of funny votes sent by the user
    "funny": 88,

    // integer, number of cool votes sent by the user
    "cool": 15,

    // integer, number of fans the user has
    "fans": 1032,
```

```
    // array of integers, the years the user was elite
    "elite": [
        2012,
        2013
    ],

    // float, average rating of all reviews
    "average_stars": 4.31,

    // integer, number of hot compliments received by the user
    "compliment_hot": 339,

    // integer, number of more compliments received by the user
    "compliment_more": 668,

    // integer, number of profile compliments received by the user
    "compliment_profile": 42,

    // integer, number of cute compliments received by the user
    "compliment_cute": 62,

    // integer, number of list compliments received by the user
    "compliment_list": 37,

    // integer, number of note compliments received by the user
    "compliment_note": 356,

    // integer, number of plain compliments received by the user
    "compliment_plain": 68,

    // integer, number of cool compliments received by the user
    "compliment_cool": 91,

    // integer, number of funny compliments received by the user
    "compliment_funny": 99,

    // integer, number of writer compliments received by the user
    "compliment_writer": 95,

    // integer, number of photo compliments received by the user
    "compliment_photos": 50
}
```

- checkin.json contains checkins on a business

```
{
    // nested object of the day of the week with key of
    // the hour (using a 24hr clock) with the count of checkins
```

```
    // for that hour (e.g. 14:00 - 14:59).
    "time": {
        "Wednesday": {
            "14:00": 2,
            "16:00": 1,
            "2:00": 1,
            "0:00": 1
        },
        "Sunday": {
            "16:00": 8,
            "14:00": 3,
            "15:00": 3,
            "13:00": 1,
            "18:00": 2,
            "23:00": 1,
            "21:00": 1,
            "17:00": 2
        },
        "Friday": {
            "16:00": 1,
            "13:00": 1,
            "11:00": 2,
            "23:00": 2
        },
    },

    // string, 22 character business id, maps to business in business.json
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg"
}
```

- tip.json - tips written by a user on a business. Tips are shorter than reviews and tend to convey quick suggestions.

```
{
    // string, text of the tip
    "text": "Secret menu - fried chicken sando is da bombbbbbb Their zapatos
are good too.",

    // string, when the tip was written, formatted like YYYY-MM-DD
    "date": "2013-09-20",

    // integer, how many likes it has
    "likes": 172,

    // string, 22 character business id, maps to business in business.json
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",

    // string, 22 character unique user id, maps to the user in user.json
```

```
    "user_id": "49JhAJh8vSQ-vM4Aourl0g"
}
```

- photos – contains entries of the following form

```
[
    {
        // string, 22 character unique photo id
        "photo_id": "_nN_DhLXkfwEkwPNxne9hw",


        // string, 22 character business id, maps to business in
business.json
        "business_id" : "tnhfDv5Il8EaGSXZGiuQGg",

        // string, the photo caption, if any
        "caption" : "carne asada fries",

        // string, the category the photo belongs to, if any
        "label" : "food"
    },
    {...}
]
```